НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

«До захисту допущено»
В.О. Завідувача кафедри
_____ О.Л. Тимощук

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Ідентифікація спаму в повідомленнях за допомогою наївного баєсового класифікатора»

Виконав:
студент IV курсу, групи КА-64
Федейко Юрій Володимирович _____

Керівник:
асистент кафедри ММСА,
Кухарєв С.О. _____

Консультант з економічного розділу:
доцент, к.е.н. Шевчук О.А.
_____

Консультант з нормоконтролю:
доцент кафедри ММСА, к.т.н.
Коваленко А.Є. _____

Рецензент:
доцент кафедри СП
Безносик О.Ю. _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____

Київ – 2020

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри
_____ Оксана ТИМОЩУК
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Федейко Юрію Володимировичу

1. Тема роботи «Ідентифікація спаму в повідомленнях за допомогою наївного баєсового класифікатора», керівник роботи Кухарєв Сергій Олександрович, затверджені наказом по університету від «_25_»_травня_ 20_20_ р. №_1143-с

2. Термін подання студентом роботи 08 травня 2020 року

3. Вихідні дані до роботи

1. Операційна система MacOs Catalina
2. Частота процесора 2.3 ГГц
3. Мова програмування Python
4. Середовище розробки – Visual Studio Code v1.45
5. Бібліотеки, що використовувалися: NumPy, Pandas, Sklearn, Matplotlib, NLTK, Time, WordCloud, Spacy

4. Зміст роботи

1. Проаналізувати існуючі підходи до вирішення даної задачі
2. Проаналізувати необхідну теорію для розробки програмного продукту
3. Розробити варіації програмного продукту для ідентифікації спаму
4. Розробити оцінки ефективності програмного продукту
5. Виконати економічний аналіз програмного продукту

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Презентація

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	к.е.н., доц. Шевчук О. А.	21.04.20	28.05.20

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	10.04.20	
2	Аналіз існуючих підходів	19.04.20	
3	Пошук дата-сетів	29.04.20	
4	Розробка програмного продукту для вирішення поставленої задачі	13.05.20	

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломної роботи.

5	Розробка системи оцінювання ефективності програмного продукту	15.05.20	
6	Тестування продукту, отримання результатів	17.05.20	
7	Оформлення дипломної роботи	19.05.20	

Студент Юрій Володимирович ФЕДЕЙКО

Керівник Сергій Олександрович КУХАРЄВ

РЕФЕРАТ

Дипломна робота: 88 с., 43 рис., 6 табл., 4 дод., 8 джерел.

ІДЕНТИФІКАЦІЯ СПАМУ В ПОВІДОМЛЕННЯХ ЗА ДОПОМОГОЮ НАЇВНОГО БАЄСОВОГО КЛАСИФІКАТОРА

В даній роботі досліджується наївний баєсовий класифікатор, його застосування для задачі ідентифікації спаму в повідомленнях та порівняння його ефективності з аналогами, у виді багатошарового перцептрона.

Метою дипломної роботи є розробка універсального програмного продукту на основі наївного баєсового класифікатора для задачі розпізнавання шкідливих повідомлень та обґрунтування його найкращої ефективності для вирішення даної задачі.

Результатом роботи є програмний продукт для ідентифікації спаму написаний мовою програмування Python та дослідження ефективності програмного продукту на основі баєсового класифікатора та його слабких місць. Використання даного програмного продукту дозволяє розпізнавати спам повідомленя на різного роду онлайн сервісах або смс повідомленнях.

ABSTRACT

Bachelor thesis: 88 p., 6 tabl., 43 fig., 4 add. and 8 references

IDENTIFICATION OF SPAM IN MESSAGES WITH HELP OF NAIVE BAYES CLASSIFIER

This work investigates the naive Bayesian classifier, its application to the problem of identifying spam in messages and comparing its effectiveness with analogues, in the form of a multilayer perceptron.

The purpose of the bachelor thesis is to develop a universal software product based on a naive Bayes classifier for the problem of recognizing malicious messages and justify its best effectiveness for solving this problem.

The result is a software product to identify spam, written in the Python programming language and research the effectiveness of the software product based on naïve bayes and it's vulnerabilities. The use of this software product allows you to recognize spam messages on various online services or SMS messages.

ЗМІСТ

ВСТУП

На сьогоднішній день весь світ переходить в цифрову сферу і в ній, попри активну боротьбу, з'являється все більше і більше різного роду зловмисників які намагаються нажитися на простих людях. Завданням яке стоїть перед моєю роботою є хоч в певній мірі запобігання поширенню різного роду атак на користувачів в мережі інтернет.

Спам це масова розсилка рекламних повідомлень, які приходять без згоди отримувача. Зазвичай такі повідомлення або листи часто містять в собі віруси. В спам-розсилці часто зустрічаються шахраї, які всіма можливими способами будуть виманювати в користувачів конфіденційну інформацію для отримання коштів незаконним способом. Своєчасне розпізнавання спаму є важливим для безпеки людей, щодня сотні тисяч людей отримують спам повідомлення.

Перший зареєстрований в історії приклад спам розсилки відбувся в 1864 році, тоді деякі британські політики отримали неочікувану телеграму, що рекламувала стоматологічні послуги. Це відбулося в наслідок того, що компанія "Western Union" добавила можливість відправки телеграм в своїй мережі зразу багатьом користувачам. Як бачимо, з збільшенням можливостей для простих людей, збільшуються можливості і для зловмисників.

Сучасні статистичні дослідження [1] показали, що на сьогодні ймовірність будь-якого повідомлення бути спамом складає 80%. Однак більшість баєсових програм розпізнавання спама роблять припущення щодо відсутності апріорних переваг у повідомленнях бути спамом, і передбачає, що у обох випадків є рівні ймовірності 50%.

Про фільтри, які використовують дану гіпотезу, говорять як про фільтри "без упереджень".

В машинному навчанні класифікацію розуміють як задачу визначення класу для нового об'єкта на основі емпіричних даних, які описують

досліджувані зразки і відображають присутні їм властивості і закономірності. Існує залежність між зразками і класами, але вона невідома. Множина прецедентів, пар зразок-клас, складає навчальну вибірку, по якій знаходиться залежність , тобто будується алгоритм здатний для будь-якого зразка видати відповідь, до якого класу він належить. Це приклад навчання з вчителем. Під вчителем в даному випадку розуміється навчальна вибірка.

Прикладами таких моделей, основаних на машинному навчанні, є баєсові класифікатори. В роботі були розглянуті наївний баєсовий класифікатор тобто поліноміальний та Бернуллі, та порівняння ефективності їх роботи з багатошаровим перцептроном.

В баєсових класифікаторах використовується критерій, що мінімізує ймовірність прийняття помилкового рішення, тому баєсові алгоритми вважаються статистично оптимальними. Однак для цього алгоритми потребують в ідеалі повного знання багатовимірних функцій розподілу спостережуваних ознак для кожного класу. Необхідність такого знання обумовлена використання формули Баєса, яка лежить в основі баєсових методів прийняття рішень.

При практичному застосуванні баєсових алгоритмів, як правило, виникають наступні проблеми. По-перше, дослідник не володіє настільки повним знанням, а по-друге, об’єм навчальної вибірки обмежений, тому не можливо емпіричним шляхом отримати інформацію про багатовимірному законі розподілу спостережуваних даних.

Для вирішення вищезгаданих проблем винайшли наївний баєсовий класифікатор, який передбачає повну відсутність статистичних зв’язків між ознаками. Таким чином, необхідність повного знання багатовимірних функцій розподілу зводиться до необхідності знання маргінальних функцій розподілу спостережуваних ознак для кожного із досліджуваних класів.

Зазвичай при побудові наївного баєсового класифікатора виходять з припущення про нормальності маргінальних функцій розподілу

спостережуваних ознак. В загальному випадку спостережувані признаки не підпорядковуються цьому розподілу.

Метою даної роботи є адаптація алгоритмів наївного баєсового класифікатора для визначення спаму в повідомленнях, порівняння ефективності різних видів наївного баєсового класифікатора та порівняння класифікатора з нейронної мережі.

Підсумовуючи все вищесказане, предметом дослідження є методи класифікації інформації за деяким ознаками її елементів, а об’єктом дослідження – наївний баєсовий класифікатор та його варіації.

1 ОСОБЛИВОСТІ КЛАСИФІКАЦІЇ БАЄСОВИМИ МЕТОДАМИ

1.1 Баєсова класифікація

На сьогоднішній день існує багато задач, для розв’язку яких застосовують класифікатори, такі як наприклад класифікація текстів по жанровим, авторським гендерним і іншим стилям чи розпізнавання семантичного забарвлення повідомлення автора. Дана теорія прийняття рішень складає основу статистичного підходу до задачі класифікації об’єктів. Цей підхід заснований на тому, що задача вибору рішення сформульована в термінах теорії ймовірності і відомі всі ймовірнісні величини, які важливі для даної задачі.

В основі даної класифікації лежить теорема Баєса. Дана теорема дозволяє визначити ймовірність будь-якої події при умові, що відбулась інша статистично пов’язана з нею подія. Іншими словами, по теоремі Баєса можна точніше перерахувати ймовірність події, врахувавши раніше відому інформацію та дані нових спостережень. Виведення теореми Баєса може бути виконане з основних аксіом теорії ймовірностей. Особливість даної теореми в тому, що для її застосування необхідна велика кількість обчислень, звідси баєсові оцінки стали активно застосовуватися тільки після розвитку обчислювальних машин [7].

Теорема Баєса:

Нехай $H_1, H_2, \ldots$ – повна група подій, і A – деяка подія, ймовірність якої додатня. Тоді умовна ймовірність того, що має місце подія H_k , якщо в результаті експерименту відбувалась подія A, може бути вирахувана по формулі:

$$P(H_k|A) = \frac{P(H_k)P(A|H_k)}{\sum_{i=1}^{\infty} P(H_i)P(A|H_i)} \quad (1.1)$$

1.2 Постановка задачі

Маємо тестову вибірку з N елементів, в нашому випадку це база даних спам повідомлень. Кожен з цих елементів належить одному з двох класів K(спам або звичайне повідомлення) і характеризується набором m числових характеристик $a_1, \dots a_m$. Нехай маємо N_k елементів з першого класу, тобто може записати вибірку наших елементів як:

$$N = \sum_{k=1}^{2} N_k \tag{1.2}$$

Значення i-го признака j-го елемента, що належить k-му класу позначимо як x_{kji}. Після таких позначень можна записати кожен елемент нашої вибірки як вектор:

$$x_{kj} = (x_{kj1}, \dots, x_{kji}, \dots, x_{kjm}) \tag{1.3}$$

Даний вектор будемо розглядати як j-ту реалізацію векторної випадкової величини ξ_k , що належить розподілу ймовірностей з щільністю $p(x_1, \dots, x_m|k)$ для кожного класу k.

Тепер спостерігаємо об’єкт для якого необхідно визначити клас, тобто спам чи звичайне повідомлення. Об’єкт характеризується тільки набором m числових ознак $x_1, \dots, x_m$. Перефразовуючи сказане вище, ми маємо речення, яке розбиваємо на слова. Кожне слово є деякою числовою ознакою. Тренуючи класифікатор, ми намагаємося знайти входження певних слів в речення і належність даного слово в більшості випадків до шкідливого спам повідомлення чи до нейтрального повідомлення. Отримавши на тренувальній вибірці масив ймовірностей ми застосовуємо класифікатор на тестовій

вибірці, де з отриманих раніше даних намагаємося визначити приналежність до певного класу нових елементів.

1.3 Загальна структура баєсового класифікатора

В основі класифікатора лежить наступне правило: класифікатор обраховує апостеріорну ймовірність $P(k|x)$ для кожного з класів K, до якого може належати досліджуваний об'єкт, і відносить даний об'єкт до апостеріорно найбільш ймовірному класу $\hat{k}$:

$$\hat{k} = \arg\max_{k} \ln P(k|\, x_1, \dots, x_m). \tag{1.4}$$

Апостеріорна ймовірність обраховується за формулою Байеса:

$$P(k|x_1, \dots, x_m) = P(k)p(x_1, \dots, x_m|k)/p(k), \tag{1.5}$$

де $P(k)$ – апріорна ймовірність того, що елемент відноситься до k-го класу;

$p(k)$ і $p(x_1, \dots, x_m|k)$ – безумовна і умовна багатомірні щільності розподілу вектора ознак, компоненти якого зазвичай статистично залежні [4].

Таким чином, баєсовий класифікатор передбачає, що багатовимірна спільна щільність розподілу ознак відома для всіх класів.

Аналітичне представлення багатовимірної щільності ймовірності відомо тільки для нормального розподілу. Разом з тим, багатовимірна нормальна щільність розподілу дає підходящу модель для одного важливого випадку, а саме коли значення векторів ознак x для даного класу k представлені неперервними, чуть викривленими версіями єдиного типового вектору, або вектора-прототипа μ_k. Саме цього очікують від класифікатора, щоб виділяти

ті признаки, які, будучи різними для образів, належні різним класам, були б можливо більш схожими для образів із одного і того ж класу.

Багатовимірна нормальна щільність розподілу в загальному виді представляється формулою:

$$p(x) = \frac{1}{(2\pi)^{\frac{m}{2}} \det R^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T R^{-1}(x-\mu)}, \quad (1.6)$$

де μ – m-компонентний вектор середнього значення;

R – матриця коваріацій розміру $m \times m$;

$T -$ знак транспонування.

Якщо ж всі недіагональні елементи дорівнюють 0, то $p(x)$ зводиться до добутку одномірних нормальних щільностей компонент вектора x.

Тому для багатовимірного нормального розподілу виходить виразити в аналітично-замкненій формі алгоритм баєсової класифікації(з точністю до незначних доданків):

$$\hat{k} = \arg\max_k (\ln P(k) - \frac{1}{2}\ln \det R_k - \frac{1}{2}(x_k - \mu_k) R_k^{-1} (x_k - \mu_k)^T), \quad (1.7)$$

де $\mu_k -$ m-вектор математичних сподівань значень ознак об'єктів класу k;

$R_k - m \times m$ матриця коваріацій векторів ознак класу k.

Діагональні елементи матриці утворюють m-вектор D_k дисперсій ознак об'єктів класу k [4].

1.4 Висновки до розділу 1

В даному розділі були описані загальні підходи до баєсових методів класифікації, описано теорему Баєса та описане поняття апостеріорної

ймовірності та поняття класифікатора, а також метод його роботи. З даного розділу випливає що методи Баєса застосовуються для задачі класифікації інформації, та дані методи можна застосувати для проблеми даної роботи, а саме ідентифікації спаму.

2 ПРИКЛАДНИЙ ПІДХІД ДО ЗАДАЧІ КЛАСИФІКАЦІЇ

2.1 Наївний баєсовий класифікатор

Наївний баєсовий класифікатор це найпростіша форма баєсової мережі, в якій всі атрибути незалежно подані числовим значенням в класі [2]. Приклад найпростішого баєсового класифікатора зображений на рисунку 2.1.

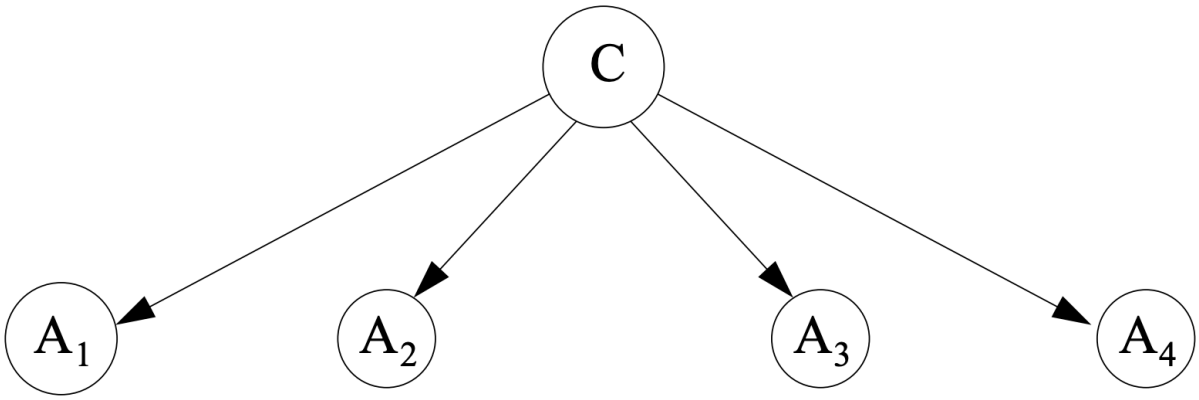

Рисунок 2.1 – Приклад наївного баєсового класифікатора

Також можливий інший варіант, аргументований наївний баєсовий класифікатор(англ. argumented Naive Bayes). Його основна відмінність в тому, що він враховує залежності між ознаками в класі (рисунок 2.2) [2].

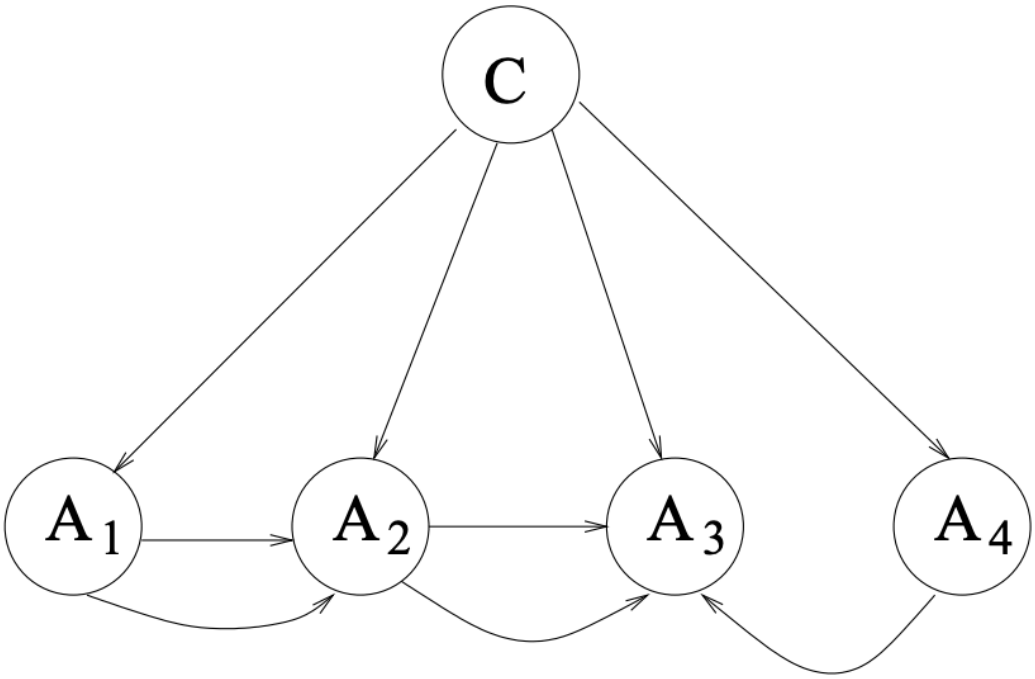

Рисунок 2.2 – Приклад аргументованого наївного баєсового класифікатора

В наївному баєсовому класифікаторі робиться припущення щодо незалежності ознак об’єкта. Якщо не брати до уваги статистичні зв’язки між компонентами вектора ознак, тоді матриця R_k буде діагональною з вектором D_k на головній діагоналі і класифікатор (1.7) стане наївним баєсовим класифікатором.

Також припускається, що маргінальна щільність розподілу $p(x_i|k)$ будь-якої ознаки є нормальною для будь-якого класу.

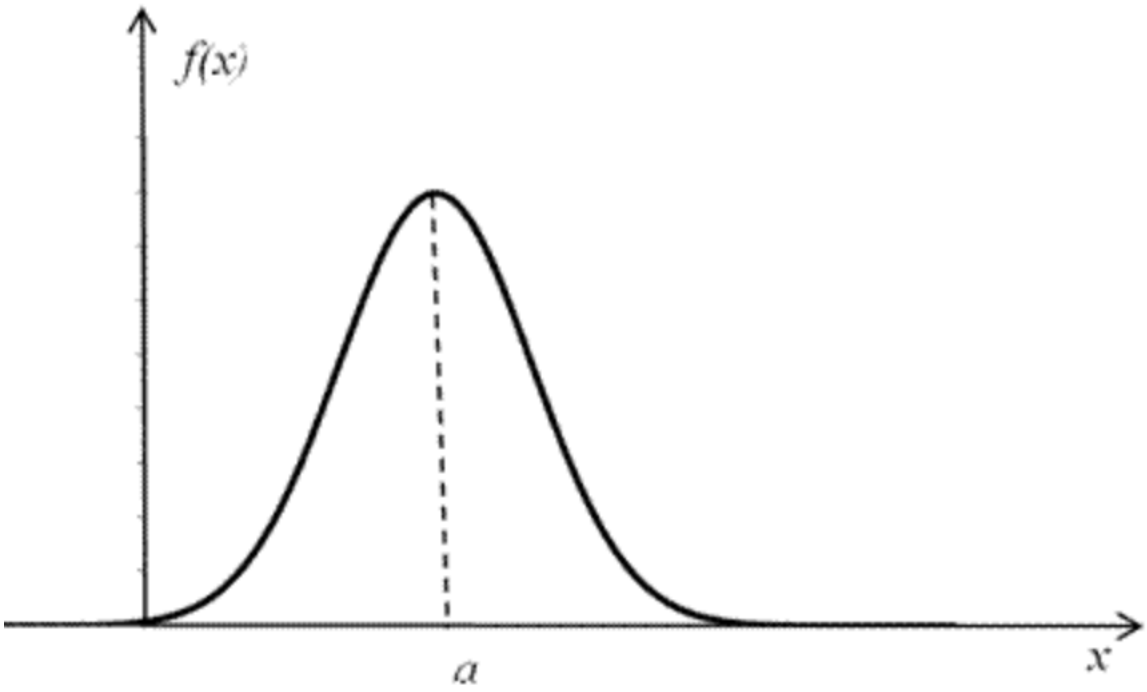

Рисунок 2.3 – Графік диференціальної функції нормального розподілу

Але на практиці так буває далеко не завжди, тобто спостережені дані не підвладні нормальному закону розподілу(рисунок 2.3), більше того, в загальному випадку закон взагалі невідомий, і має місце статистична залежність, тому область застосування класифікатора звужується.

Наївний баєсовий класифікатор має декілька варіацій для даних з різними типом функції розподілу. Для задач класифікації документів та роботи з текстом найчастіше використовують поліноміальний та Бернуллі

2.1.1 Поліноміальний наївний баєсовий класифікатор

З поліноміально розподіленою моделлю, вектори ознак представляють частоти з яким генеруються певні події $(p_1, \dots, p_n)$, де p_i ймовірність що подія і відбудеться. Вектор ознак $x = (x_1, \dots, x_n)$ тоді є гістограмою, де x_i сума всіх разів коли подія і зустрічалася в навчальній вибірці.

Цей класифікатор часто використовують для класифікації документів, з подіями що означають частоту появи слова в одному документі [3]. Апостеріорна ймовірність тоді обраховується за формулою:

$$p(x|k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}^{x_i} \tag{2.1}$$

Якщо заданий клас і ознака раніше не зустрічалися разом в навчальній вибірці, тоді класифікатор автоматично призначить ймовірність 0, тому що ймовірність прямо пропорційна кількості появ ознаки(в нашому випадку слова) в навчальній вибірці. Це проблематично, оскільки при обрахунку ймовірності елемента вибірки належати тому чи іншому класу, ймовірності перемножуються і неможливо зробити вірний прогноз. Це явище відоме під назвою «нульова частота». Дана проблема вирішується за допомогою згладжування по Лапласу і тоді формула для ймовірності ознаки і:

$$p_i = \frac{x_i + \alpha}{N + \alpha n}$$

Якщо $\alpha = 1$, то це згладжування по Лапласу, а якщо $\alpha < 1$ то це згладжування Лідстоуна.

2.1.2 Наївний баєсовий класифікатор Бернуллі

Наївний баєсовий класифікатор Бернуллі найкраще підходить для класифікації даних, розподіл яких підпадає під розподіл Бернуллі. В багатовимірній моделі подій Бернуллі признаками є незалежні логічні (двійкові) змінні, що описують вхідні дані. Як і поліноміальна модель, ця

модель популярна для задач класифікації документів, де використовуються двійкові признаки входження терміну, а не частоти терміну. Якщо x_i логічне значення що виражає наявність або відсутність і-го терміну в словнику, то ймовірність документа, заданого класом, виражається формулою:

$$\mathrm{p(x|k)} = \prod_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{p_{ki}^{x_i}} \, (1 - \mathrm{p_{ki}})^{1-\mathrm{x_i}}, \qquad (2.2)$$

де p_{ki} це ймовірність класу k містити вираз x_i [3].

Дана модель особливо популярна і ефективна для класифікації коротких текстів.

2.2 Статистична міра tf-idf

Term Frequency-Inverse Document Frequency (TF-IDF) − статистична міра, використовується для оцінки важливості слова в контексті деякого документа, що в свою чергу є частиною колекції документів. Це простий і зручний спосіб оцінити важливість терміну для будь-якого документа відносно всіх інших документів.

Принцип такий: якщо слово зустрічається в якому-небудь документі часто, при цьому зустрічаючись рідко у всіх інших документах − це слово має велику значимість для того самого документа.

В першій частині TF вимірює, наскільки часто термін зустрічається в документі. Логічно припустити, що в довгих документах термін може зустрітися в більших кількостях, ніж в коротких, тому абсолютні числа ту не підходять. Тому застосовують відносні – ділять кількість раз, коли потрібний термін зустрівся в тексті на загальну кількість слів в тексті.

Тобто якщо інтерпретувати на сказане вище про баєсові класифікатори, то:

$$\mathrm{tf(a)} = \frac{\mathrm{n_a}}{\mathrm{n_i}}, \tag{2.3}$$

де $\mathrm{n_a}$ кількість повторень ознаки "a" в елементі вибірки;

$\mathrm{n_i}$ – загальна кількість ознак в елементі i.

IDF – це обернена частота документів. Вони визначає безпосередньо важливість терміну. Тобто, коли ми обраховували TF, всі терміни вважалися ніби рівними по важливості один до одного. Але відомо що прийменники зустрічаються дуже часто, хоча практично не впливають на сенс тексту.

$$\mathrm{idf(a)} = \log \frac{\mathrm{N_D}}{\mathrm{N_a}}, \tag{2.4}$$

де $\mathrm{N_D}$ – загальна кількість елементів вибірки;

$\mathrm{N_a}$ – кількість елементів вибірки які містять в собі ознаку "a".

В кінці кінців перемноживши ці дві величини отримали значення tf-idf для всіх ознак у всіх елементах. В своїй роботі я користувався даною оцінкою для роботи з класифікаторами та нейронною мережею.

2.3 Недоліки наївного баєсового класифікатор

Наївний баєсовий класифікатор має ряд недоліків які в кінцевому результаті впливають на точність моделі. В своїй роботі я намагався мінімізувати їх вплив.

2.3.1 Проблема нечастих слів

Дана проблема виникає в випадку, якщо слово ніколи не зустрічалося під час фази навчання: і чисельник і знаменник дорівнюють нулю, і в загальній і в формулі “спамовитості”:

$$\Pr(S|W) = \frac{\Pr(W|S)}{\Pr(W|S)+\Pr(W|H)}, \quad (2.5)$$

де $\Pr(S|W)$ – називається “спамовитістю” слова W;

При цьому число $\Pr(W|S)$, використане в наведеній вище формулі, приближено дорівнює відносній частоті повідомлень, що містять в собі слово W в повідомленнях, ідентифікованих як спам під час фази навчання.

Точно так само $\Pr(W|H)$ приближено дорівнює відносній частоті повідомлень, що містять слово W в повідомленнях, ідентифікованих як “ham” під час фази навчання.

В цілому, слова з якими програма стикнулася тільки декілька разів під час фази навчання не є репрезентативними(набір даних в вибірці занадто малий для того щоб зробити надійний висновок про властивості такого слова). Найпростіше рішення полягає в тому, щоб просто ігнорувати такі ненадійні слова.

2.3.2 Нейтральні слова

Нейтральними називаються слова по типу “the”, “a”, ”some”, “is” і подібні в англійській мові, або їх еквіваленти в інших мовах. Ці слова мають

бути проігноровані, оскільки не несуть в собі ніякого смислового навантаження.

Деякі баєсові фільтри взагалі просто ігнорують всі слова, “спамовитість” яких близько 0.5, оскільки в цьому випадку виходять якісно кращі результати. Враховуються тільки ті слова, “спамовитість” яких близько 0.0 (ознака законних повідомлень), або поруч з 1.0 (ознака спама).

Деякі програмні продукти можуть брати до уваги той факт, що деяке слово повторюється декілька раз в перевіряємому повідомленні, інші цього не роблять.

Також деякі продукти можуть використовувати словосполучення замість конкретних слів. В деяких випадках це може підвищити точність.

2.4 Нейронна мережа як метод вирішення задачі ідентифікації спама

У вирішенні задачі класифікації спама я не обмежився застосуванням тільки наївного баєсового класифікатора. Для порівняння результатів та в загальному ефективності роботи я вибрав багатошаровий перцептрон. Його також деколи використовують для задач класифікації.

Ідею перцептрона запропонував нейрофізіолог Френк Розенблатт. Він запропонував схему, що моделювала процес людського сприйняття.

В загальному випадку перцептрон складається з трьох основних елементів: вхід, прихований шар і вихід (рисунок 2.4).

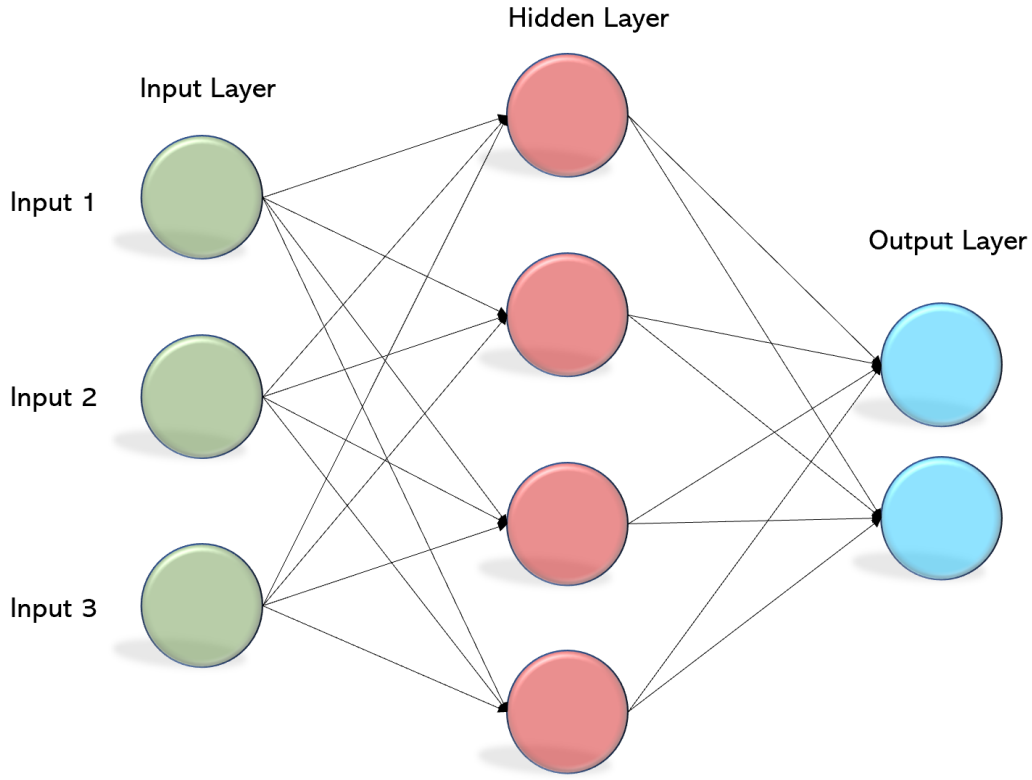

Рисунок 2.4 – Модель багатошарового перцептрона

Основним завдання нейронної мережі є знаходження вагових коефіцієнтів на зв’язках між елементами, щоб після навчання при поданні якогось сигналу на вхід, мережа могла дати чіткий вихід.

Одною з основних задач з якою справляється багатошаровий перцептрон є задача класифікації, вона зображена на рисунку 2.5.

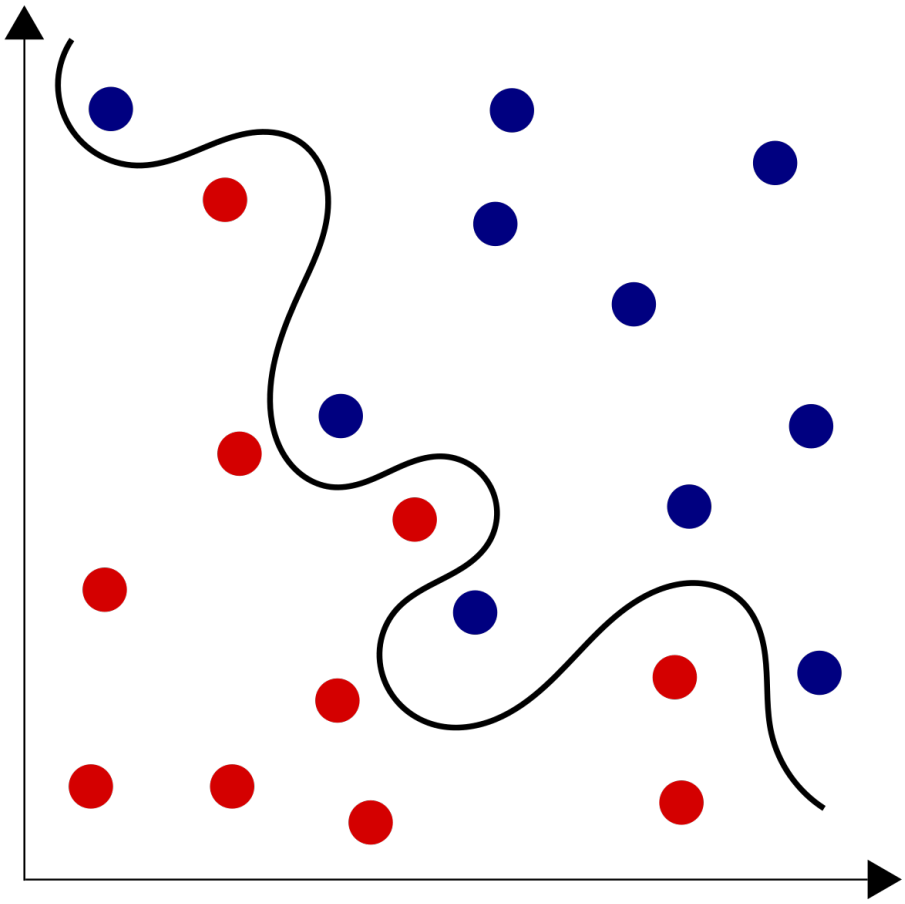

Рисунок 2.5 – Класифікація багатошаровим перцептроном

Як бачимо на рисунку п’ять є два класи, і по координатам x, y перцептрон навчився визначати до якого класу належить елемент. Це досить схоже на мою задачу, розпізнавання спаму, де в ролі двох класів є спам або звичайне повідомлення. Для розв’язку моєї задачі я використав три приховані шари [4].

Алгоритм навчання перцептрона:

1. Присвоїти ваговим коефіцієнтам $w_1, w_2, \ldots, w_n$ деякі початкові значення;

2. Подати вхідний образ X і порахувати вихід (Y). Якщо вихід правильний то перейти до кроку 4, в іншому випадку до кроку 3;

3. Виконати корегування вагових коефіцієнтів:

- якщо вихід невірний і Y = 0, то необхідно збільшити вагові коефіцієнти тих входів, на які була подана одиниця;

- якщо вихід невірний і Y = 1, то необхідно зменшити вагові коефіцієнти тих входів, на які була подана одиниця;

4. Повторювати кроки 2-4 даного алгоритму навчання перцептрона доки мережа не почне видавати очікуваний результат на векторах з навчальної вибірки або поки відхилення не стане нижче деякого значення.

На вхід багатошарового перцептрона подається tf-idf вектор, описаний вище.

2.5 Висновки до розділу 2

В даному розділі описується прикладний підхід до задачі класифікації наївним баєсовим класифікатором, тобто перехід від теорії до практичних варіацій баєсового класифікатора з усіма недоліками та проблемами які можуть виникнути в процесі навчання. Також в даному розділі було розглянуто аналог для вирішення даної задачі – нейронну мережу, а саме багатошаровий перцептрон. Було розписано статистичну tf-idf міру, необхідну для подальшого представлення вхідних повідомлень для навчання як класифікатора, так і нейронної мережі.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ІДЕНТИФІКАЦІЙ СПАМУ

3.1 Аналіз дата-сетів

Для даної роботи було вибраний два з небагатьох доступних дата-сетів базою спамових смс повідомлень “SMS Spam Collection v.1” з інтернет ресурсу ResearchGate [6] та власно створений дата-сет з коментарів на онлайн сервісі “YouTube”. Перший містить близько п’яти з половиною тисяч різного роду повідомлень, а другий – двох тисяч. Зараз по черзі проведу коротку характеристику даних з обох дата-сетів.

Обоє дата-сетів приведені до вигляду файлів з двома колонками: “category” і “text”, в яких відповідно спочатку написаний тип повідомлення(спам або звичайне повідомлення) та сам зміст повідомлення (рисунок 3.1).

category	text
0	go jurong point crazi avail bugi n great world...
0	ok lar joke wif u oni
1	free entri 2 wkli comp win fa cup final tkts 2...
0	u dun say earli hor u c alreadi say
0	nah dont think goe usf live around though
1	freemsg hey darl 3 week word back id like fun ...
0	even brother like speak treat like aid patent
0	per request mell mell oru minnaminungint nurun...
1	winner valu network custom select receivea ï¿½...
1	mobil 11 month u r entitl updat latest colour ...
0	im gonna home soon dont want talk stuff anymor...
1	six chanc win cash 100 20000 pound txt csh11 s...
1	urgent 1 week free membership ï¿½100000 prize ...
0	ive search right word thank breather promis wo...
0	date sunday

Рисунок 3.1– Вигляд першого дата-сету з спам повідомленнями

Для першого дата-сету:

- Повідомлень з спамом в дата-сеті 13.41% і відповідно 86.59% звичайних повідомлень (рисунок 3.2).

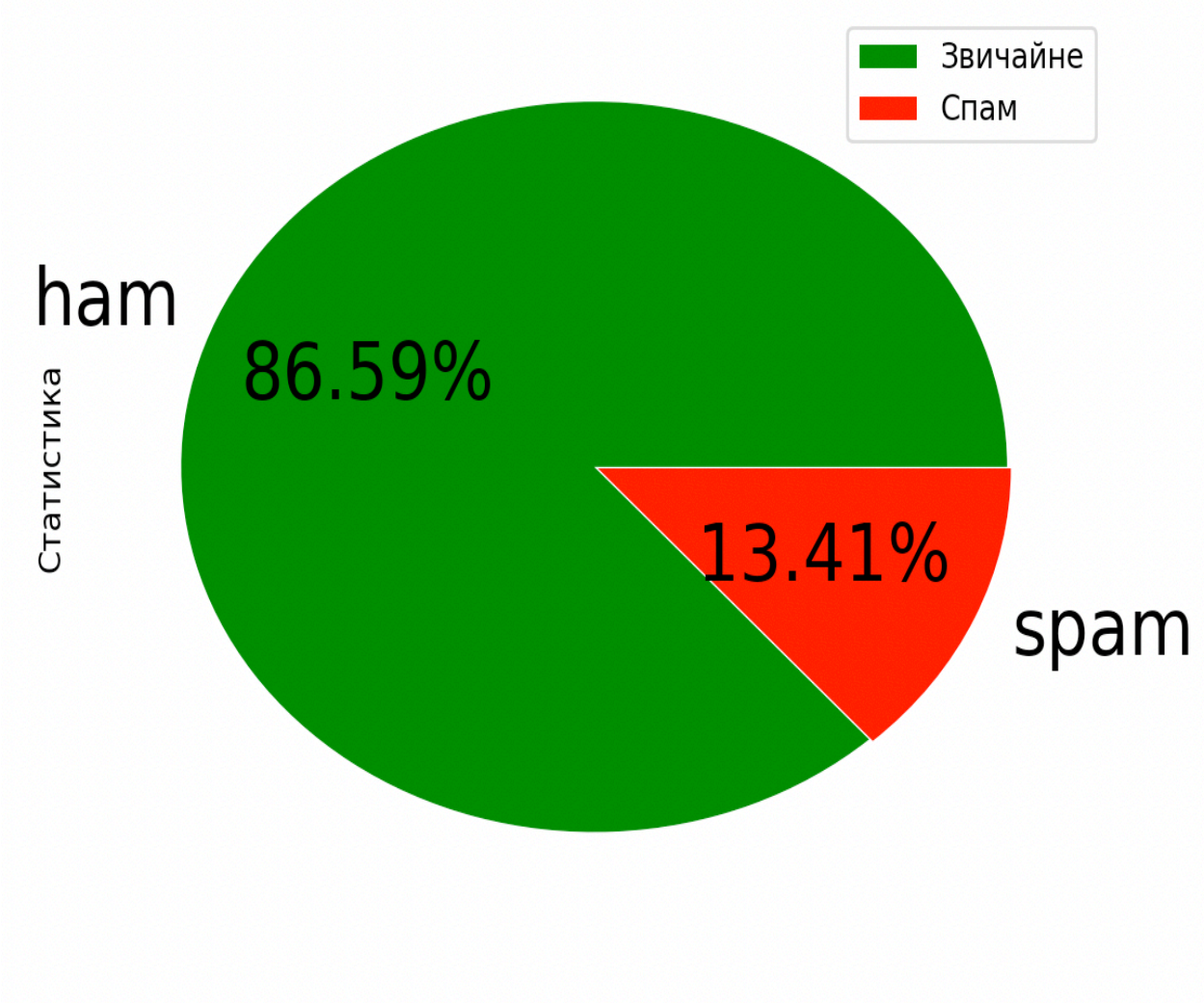

Рисунок 3.2 – Відсоткове співвідношення звичайних і спам повідомлень

- Зображення всіх слів які зустрічаються в спам (рисунок 3.3) та звичайних (рисунок 3.4) повідомленнях (чим більше слово тим частіше воно зустрічається).

Рисунок 3.3 – Слова які властиві спам повідомленням

Рисунок 3.4 – Слова які властиві звичайним повідомленням

Можна побачити, що у спам повідомленнях найпопулярніші слова “free, call” тобто “безплатно, подзвони” (рисунок 3.3, рисунок 3.4). Це логічно, оскільки найчастіші спам повідомлення пов’язані з різного роду безплатними благами, і щоб їх отримати потрібно подзвонити чи перейти на певного роду ресурси.

- Порівняємо довжини спам і звичайних повідомлень (рисунок 3.5 та рисунок 3.6).

Рисунок 3.5 – Довжина спам повідомлень

Рисунок 3.6 – Довжина звичайних повідомлень

Як бачимо, по статистиці, спам повідомлення зазвичай довші за звичайні.

Тепер розглянемо другий дата-сет:

В цьому випадку кількість звичайних і спам повідомлень майже однакова (рисунок 3.7).

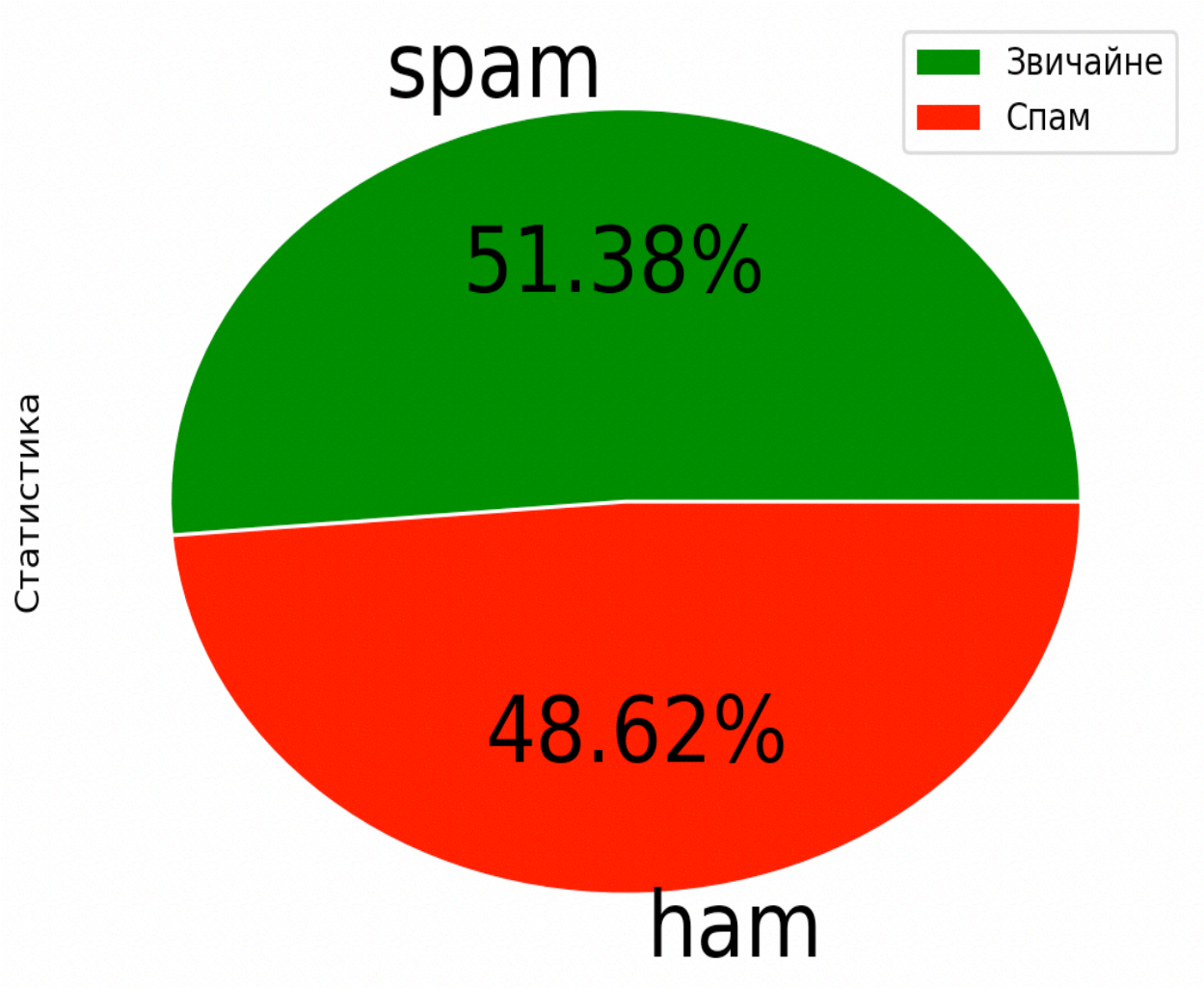

Рисунок 3.7 – Співвідношення звичайних і спам коментарів в другому дата-сеті

- Далі розглянемо найчастіші слова властиві спам та звичайним коментарям (рисунок 3.8, рисунок 3.9).

Рисунок 3.8 – Найчастіші слова в спам коментарях

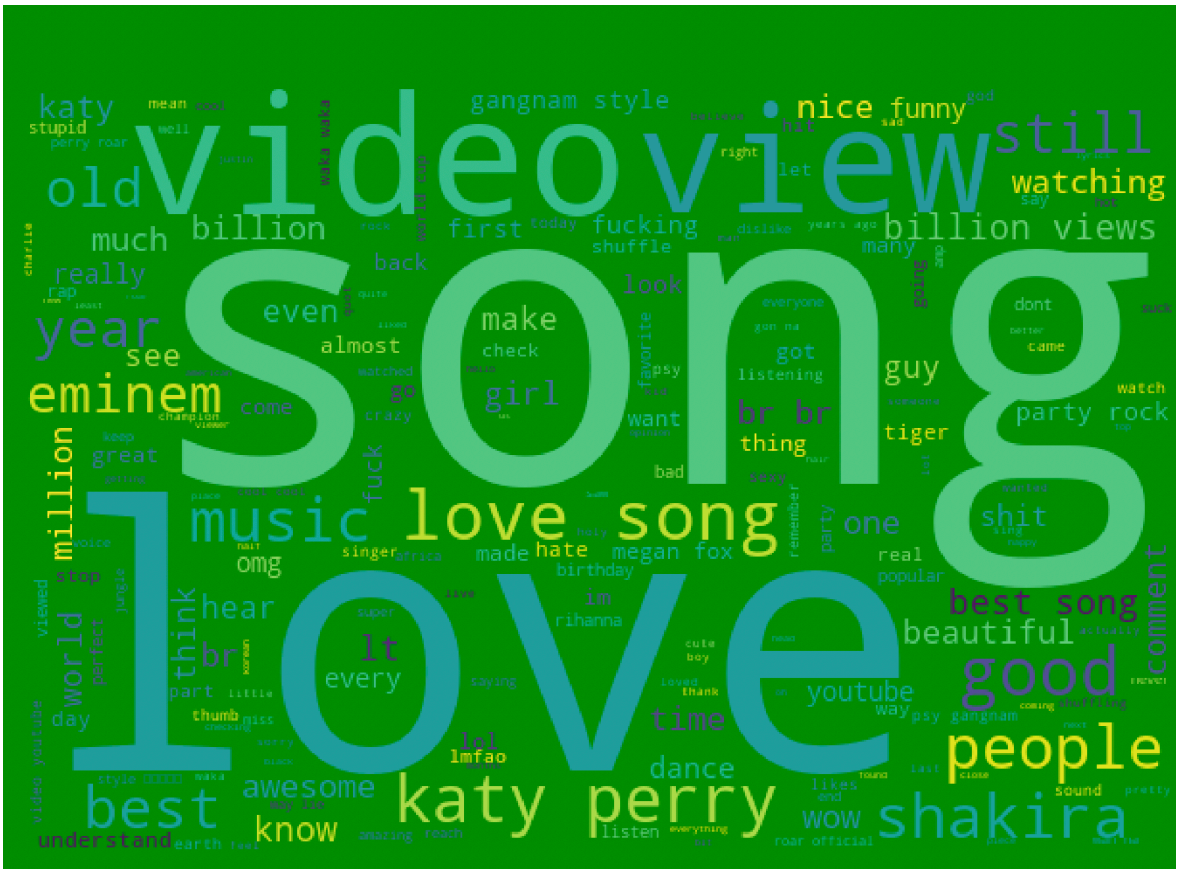

Рисунок 3.9 – Найчастіші слова в нейтральних коментарях

Оскільки дата-сет досить невеликий, близько двох тисяч коментарів, то неможливо досить чітку картину відслідкувати, але по спам все таки видно що багато використовують слово “check” та “subscribe”, які перекладаються як “перевір” та “підпишись”. З першим словом зазвичай пов’язані посилання на інші ресурси, а з другим особисті канали зловмисників, де міститься шкідлива інформація. Щодо нейтральних коментарів, то це очікувано що слово “song” тобто “пісня” одне з найпопулярніших, оскільки даний дата-сет зібраний з коментарів під відомими музичними кліпами.

Порівняємо довжини спам і звичайних повідомлень (рисунок 3.10, рисунок 3.11).

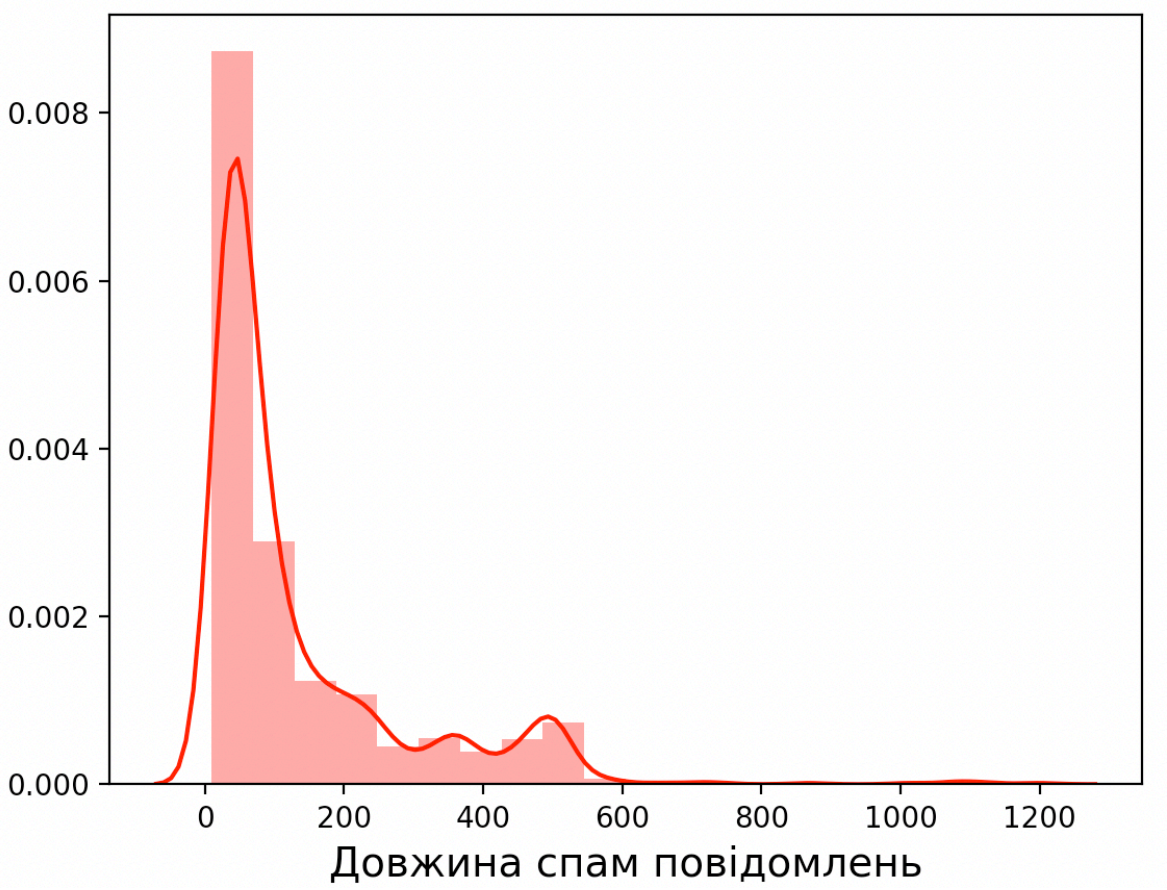

Рисунок 3.10 – Довжина спам коментарів

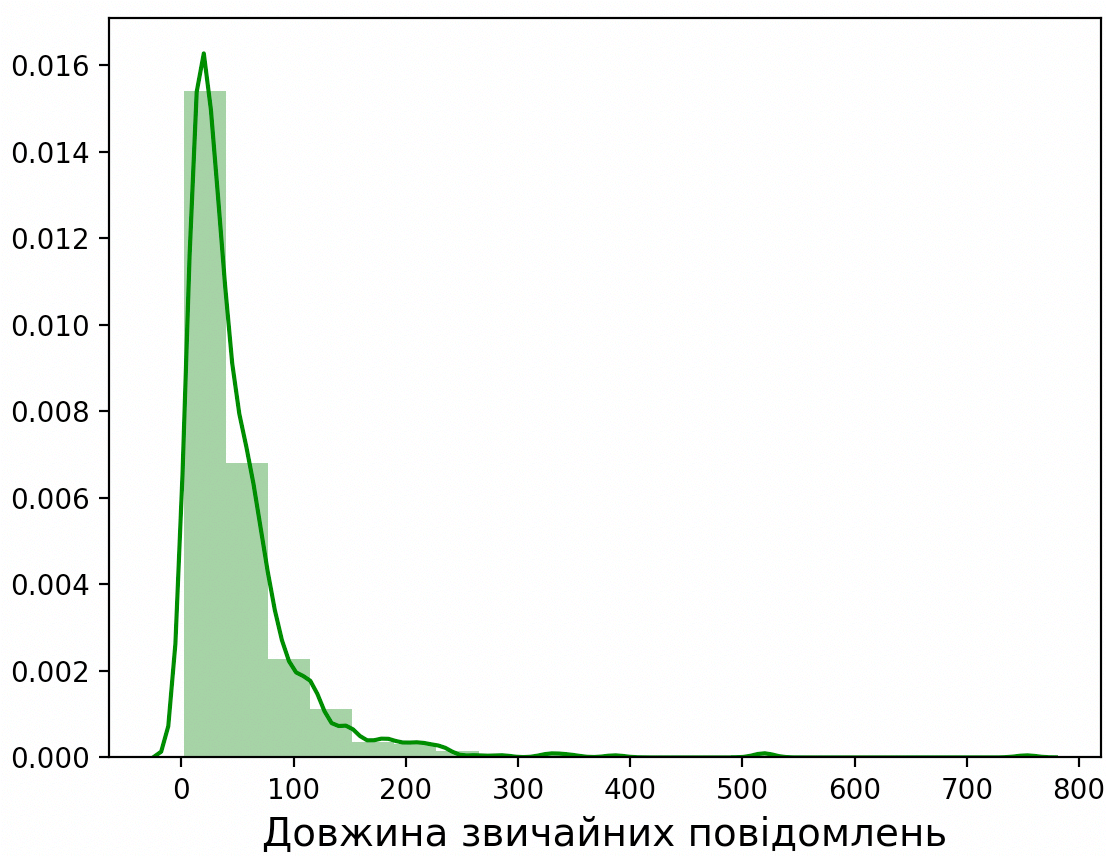

Рисунок 3.11 – Довжина звичайних коментарів

3.2 Експерименти

В своїй роботі я порівнював дві варіації наївного баєсового класифікатора (поліноміального та Бернуллі) а також порівняв наївний баєсовий класифікатор з одною з найпростіших нейронних мереж для задачі розпізнавання спама в повідомленнях.

Оскільки перший дата-сет має більший об’єм, а для такого роду задач це дуже важливо, то більшість порівняльних тестів між моделями проведу на ньому і в кінці порівняю результати моделі для різних дата-сетів щоб побачити залежність результатів від об’єму даних.

Почнемо з первинної обробки повідомлень, тобто зведення всіх слів до маленької літери, видалення розділових знаків. Далі необхідно позбутися від нейтральних слів (рисунок 3.12).

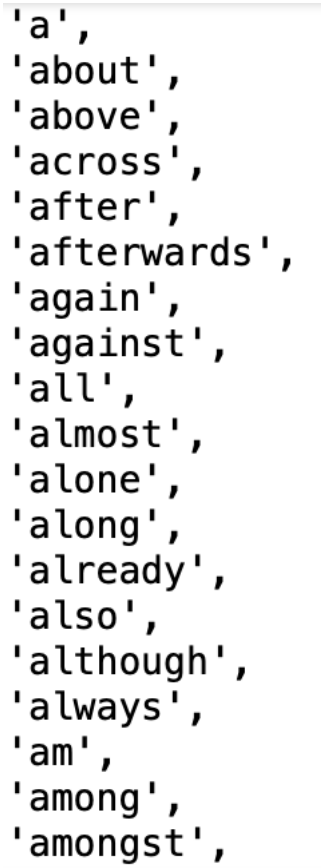

Рисунок 3.12 – Нейтральні слова для баєсового класифікатора в англійській мові

Далі йде формування оцінки tf-idf (рисунок 3.13) для подальшого використання для навчання і розпізнавання спам повідомлень.

```
(0, 7499)     0.19645294296504998
(0, 1180)     0.35363478165326045
(0, 3334)     0.1661310925744238
(0, 2032)     0.29875209257436797
(0, 1756)     0.33758262412831513
(0, 4185)     0.29875209257436797
(0, 7691)     0.2403217036843895
(0, 3372)     0.19645294296504998
(0, 1758)     0.29875209257436797
(0, 1364)     0.2689106500737722
(0, 2245)     0.27386579528069094
(0, 5494)     0.24146930553432594
(0, 4045)     0.35363478165326045
(1, 5161)     0.5633086751818669
(1, 7606)     0.44480400570972006
(1, 4011)     0.47731294876998304
(1, 4220)     0.42078899608869724
(1, 5132)     0.2827396376113674
(2, 71)       0.23721407928875096
(2, 1248)     0.1696767395440564
(2, 5767)     0.23721407928875096
(2, 7204)     0.12747363739866063
(2, 5723)     0.23721407928875096
(2, 5807)     0.16314375518709073
```

Рисунок 3.13 – Tf-idf представлення дата-сета

Як бачимо, зліва пара чисел: це номер елемента вибірки (повідомлення) та унікальний токен слова. Число справа, це пораховане tf-idf значення, себто важливість слова в тексті.

Після цього наступає навчання на тренувальній вибірці і прогноз. На рисунку 3.14 зображено тільки перших сімдесят п’ять елементі з тестової вибірки та їх апостеріорні ймовірності (зеленим ймовірність повідомлення бути спамом і оранжевим – звичайним). Яка з цих двох ймовірностей більша, до такого класу (спам або звичайне) і відноситься повідомлення.

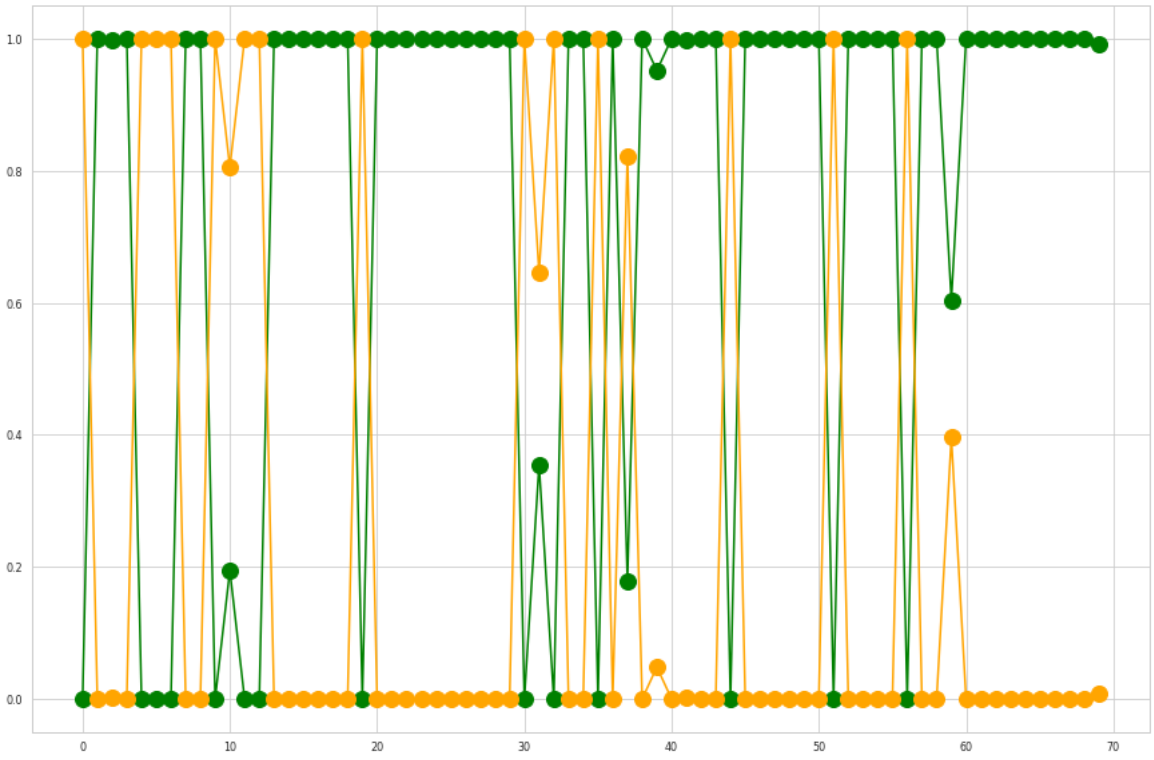

Рисунок 3.14 – Графік апостеріорних ймовірностей для поліноміального наївного баєса

По самій моделі отримано наступні результати (рисунок 3.15).

```
Поліноміальний наївний баєсовий класифікатор
Точність моделі:  0.9641062455132807
Час затрачений на навчання та прогнозування:  0.004263877868652344
```

Рисунок 3.15 – Результат поліноміальної моделі (Експеримент 1)

Як бачимо, точність прогнозування складає 96.4% що є дуже хорошим показником, враховуючи час затрачений на навчання і прогнозування.

Далі та ж сама задача тільки для наївного баєсового класифікатора Бернуллі (рисунок 3.16).

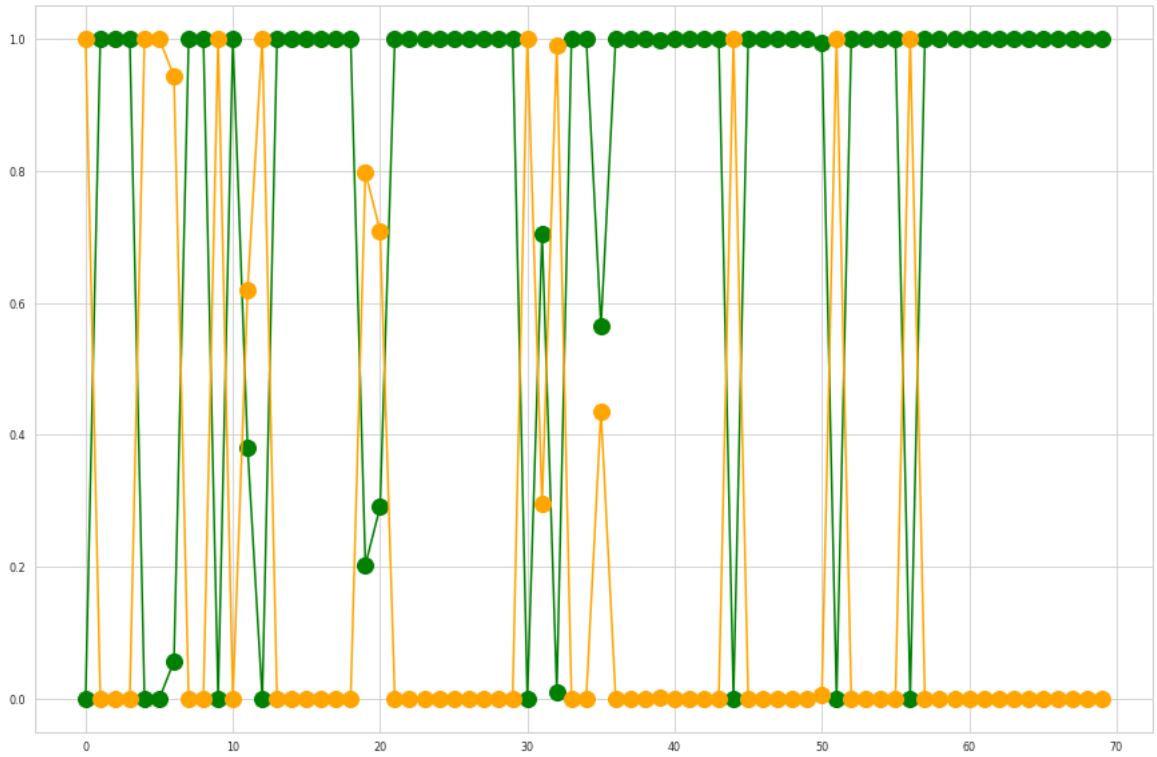

Рисунок 3.16 – Графік апостеріорних ймовірностей для наївного баєса Бернуллі

З графіків видно, що вони досить схожі.

З результатів видно, що для даного дата-сету модель Бернуллі (рисунок 3.17) підходить краще, тобто точність розпізнавання 97.4%, але час чуть більший.

```
Наївний баєсовий класифікатор Бернуллі
Точність моделі:  0.9741564967695621
Час затрачений на навчання та прогнозування:  0.0050830841064453125
```

Рисунок 3.17 – Результат моделі Бернуллі (Експеримент 1)

Оскільки кожен раз генерується випадкова вибірка з тестових та навчальних елементів, проведемо ще експериментів щоб точно переконатися в оптимальності використання моделі Бернуллі перед поліноміальною, та щоб побачити середній час виконання.

Експеримент 2 (рисунок 3.18, рисунок 3.19).

```
Поліноміальний наївний баєсовий класифікатор
Точність моделі:  0.964824120603015
Час затрачений на навчання та прогнозування:  0.004391908645629883
```

Рисунок 3.18 – Експеримент 2 для поліноміальної моделі

```
Наївний баєсовий класифікатор Бернуллі
Точність моделі:  0.9734386216798278
Час затрачений на навчання та прогнозування:  0.004587888717651367
```

Рисунок 3.19 – Експеримент 2 для моделі Бернуллі

Експеримент 3 (рисунок 3.20, рисунок 3.21).

```
Поліноміальний наївний баєсовий класифікатор
Точність моделі:  0.9748743718592965
Час затрачений на навчання та прогнозування:  0.0035028457641601562
```

Рисунок 3.20 – Експеримент 3 для поліноміальної моделі

```
Наївний баєсовий класифікатор Бернуллі
Точність моделі:  0.9748743718592965
Час затрачений на навчання та прогнозування:  0.0040283203125
```

Рисунок 3.21 – Експеримент 3 для моделі Бернуллі

Експеримент 4 (рисунок 3.22, рисунок 3.23).

```
Поліноміальний наївний баєсовий класифікатор
Точність моделі:  0.9712849964106246
Час затрачений на навчання та прогнозування:  0.0034058094024658203
```

Рисунок 3.22 Експеримент 4 для поліноміальної моделі

```
Наївний баєсовий класифікатор Бернуллі
Точність моделі:  0.9763101220387652
Час затрачений на навчання та прогнозування:  0.004065036773681641
```

Рисунок 3.23 – Експеримент 4 для моделі Бернуллі

Експеримент 5 (рисунок 3.24, рисунок 3.25).

```
Поліноміальний наївний баєсовий класифікатор
Точність моделі:  0.9720028715003589
Час затрачений на навчання та прогнозування:  0.003072977066040039
```

Рисунок 3.24 – Експеримент 5 для поліноміальної моделі

```
Наївний баєсовий класифікатор Бернуллі
Точність моделі:  0.9755922469490309
Час затрачений на навчання та прогнозування:  0.003979921340942383
```

Рисунок 3.25 – Експеримент 5 для моделі Бернуллі

Як бачимо, практично у всіх експериментах наївний баєсовий класифікатор Бернуллі показав себе краще, але за гірший час.

Середній час та середня точність для поліноміального:

$$T_{сП} = \frac{\sum_{n=1}^{5} t_n}{n} = 0.00371, \quad (3.1)$$

де t_n – час навчання і розпізнавання для кожного експерименту.

$$AC_{сП} = \frac{\sum_{n=1}^{5} ac_n}{n} = 96.9\%, \quad (3.2)$$

де ac_n – точність моделі для кожного експерименту.

Середній час та точність для Бернуллі:

$$\mathrm{T_{сБ}} = \frac{\sum_{n=1}^{5} \mathrm{t_n}}{\mathrm{n}} = \ 0.00434, \tag{3.3}$$

де $\mathrm{t_n}$ – час навчання і розпізнавання для кожного експерименту.

$$\mathrm{AC_{сБ}} = \frac{\sum_{n=1}^{5} \mathrm{ac_n}}{\mathrm{n}} = \ 97.5\%, \tag{3.4}$$

де ac_n – точність моделі для кожного експерименту.
Різниця в часі виконання є, але вона не критична.

Тепер коли ми вибрали кращою модель Бернуллі, порівняємо ефективність двох підходів: класифікатор та нейронна мережа. Мною був вибраний багатошаровий перцептрон, на вхід якого подавалася все та ж оцінка tf-idf і ось які результати були отримані (рисунок 3.26).

```
Багатошаровий перцептрон
Точність моделі:  0.9763101220387652
Час затрачений на навчання та прогнозування:  33.62564730644226
```

Рисунок 3.26 – Експеримент 1 для перцептрона

З першого ж разу він показав точність чуть кращу ніж середня для наївного баєса Бернуллі, але час навчання приблизно 7500 раз більший ніж для Бернуллі. У випадку нейронної мережі не буде графіка апостеріорних ймовірностей, тільки графік помилки(тобто різниці між отриманим результатом на виході і очікуваним результатом) зображено на рисунку 3.27.

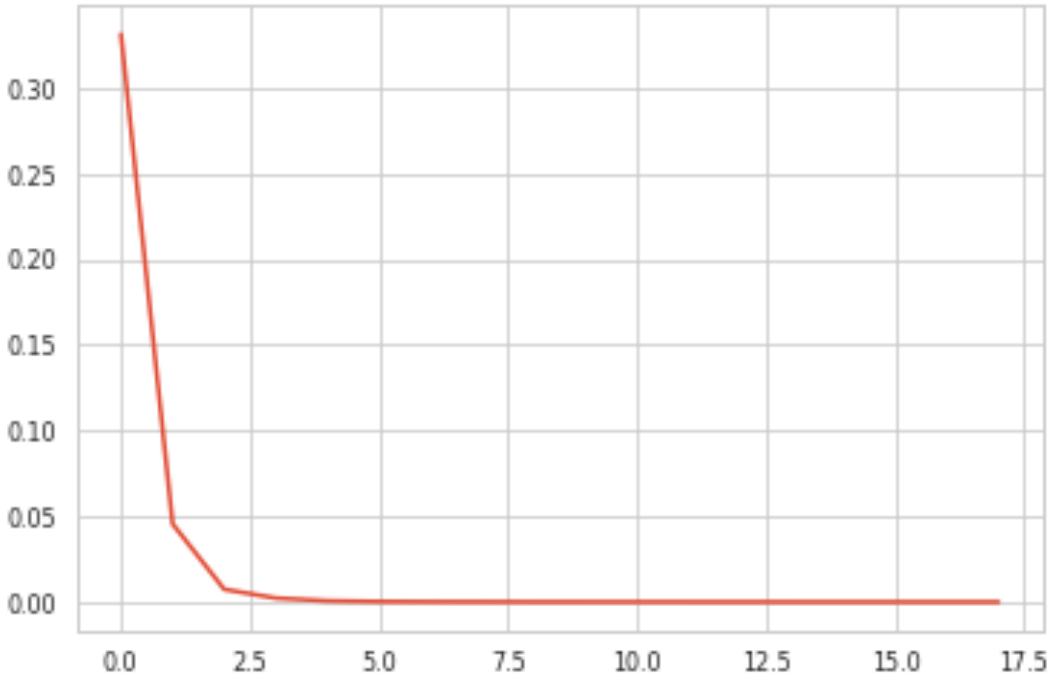

Рисунок 3.27 – Графік помилки при навчанні перцептрона

Час навчання такий великий оскільки нейронній мережі знадобилося 18 ітерацій щоб налаштувати свої вагові коефіцієнти на розпізнавання спам повідомлень.

Проведемо ще декілька експериментів щоб перевірити середній час на навчання і середню точність.

Експеримент 2 (риунок 3.28)

```
Багатошаровий перцептрон
Точність моделі:  0.9842067480258435
Час затрачений на навчання та прогнозування:  34.87250804901123
```

Рисунок 3.28 – Експеримент 2 для перцептрона

Експеримент 3 (рисунок 3.29)

```
Багатошаровий перцептрон
Точність моделі:  0.9798994974874372
Час затрачений на навчання та прогнозування:  34.36840510368347
```

Рисунок 3.29 – Експеримент 3 для перцептрона

Експеримент 4 (рисунок 3.30)

```
Багатошаровий перцептрон
Точність моделі:  0.9849246231155779
Час затрачений на навчання та прогнозування:  34.38524794578552
```

Рисунок 3.30 – Експеримент 4 для перцептрона

Експеримент 5 (рисунок 3.31)

```
Багатошаровий перцептрон
Точність моделі:  0.9770279971284996
Час затрачений на навчання та прогнозування:  37.16425013542175
```

Рисунок 3.31– Експеримент 5 для перцептрона

З проведених експериментів видно, що точність моделі краща, але час значно гірший.

Середній час та точність для перцептрона:

$$T_{сПе} = \frac{\sum_{n=1}^{5} t_n}{n} = 34.88, \quad (3.5)$$

де t_n – час навчання і розпізнавання для кожного експерименту.

$$\mathrm{AC_{сПе}} = \frac{\sum_{n=1}^{5} \mathrm{ac_n}}{n} = 98.0\%, \quad (3.6)$$

де $\mathrm{ac_n}$ – точність моделі для кожного експерименту.

Тепер, проаналізувавши роботу класифікатора на нейронної мережі для задачі розпізнавання спаму, перевірю роботу цих моделей для дата-сету з значно меншою кількістю даних.

Результат поліноміальної моделі (рисунок 3.32).

```
Поліноміальний наївний баєсовий класифікатор
Точність моделі:  0.8261758691206544
Час затрачений на навчання та прогнозування:  0.002505064010620117
```

Рисунок 3.32 – Результат роботи поліноміального наївного баєсового класифікатора для дата-сету з коментарями

Результат наївного баєсового класифікатора Бернуллі (рисунок 3.33).

```
Наївний баєсовий класифікатор Бернуллі
Точність моделі:  0.8670756646216768
Час затрачений на навчання та прогнозування:  0.004046201705932617
```

Рисунок 3.33 – Результат роботи наївного баєсового класифікатора Бернуллі для дата-сету з коментарями

Результат багатошарового перцептрона (рисунок 3.34), для його навчання знадобилося 34 ітерації.

```
Багатошаровий перцептрон
Точність моделі:  0.8752556237218814
Час затрачений на навчання та прогнозування:  12.963635921478271
```

Рисунок 3.34 – Результат роботи багатошарового перцептрона для дата-сету з коментарями

Можемо спостерігати, що в загальному точність розпізнавання впала для всіх моделей. Це пояснюється об’ємом дата-сету, він занадто малий. По часу затраченому для навчання та прогнозування спостерігаємо такий же результат як і раніше, що серед класифікаторів найдовше працює класифікатор Бернуллі і перцептрон працює в тисячі разів довше за класифікатори, але точність розпізнавання там трохи вища.

3.3 Висновки до розділу 3

Враховуючи всі отримані результати, можна сказати що наївний баєсовий класифікатор Бернуллі є найоптимальнішим для вирішення задачі класифікації спам повідомлень, оскільки дана задачу буде стояти перш за все в різного роду месенджерах та онлайн платформах, де спам атаки розгортаються досить швидко і щоб їм протистояти, тобто блокувати на етапі початку атаки, потрібно швидко перелаштовувати модель під нові слова. З навчання нейронної мережі з більшою кількістю повідомлень це стане робити все складніше і складніше, а наївний баєсовий класифікатор переналаштовує свої апостаріорні ймовірності дуже швидко, тому зможе динамічно обробляти потік спам повідомлень і переналаштовуватися під нові атаки.

Також робимо висновок що об’єм бази даних для даного роду задач є критично важливим, бо прямо пропорційно впливає на результат розпізнавання.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі оцінюється програмний продукт для розпізнавання спам повідомлень. Програму було реалізовано мовою Python з використанням бібліотек NumPy, SciPy, Pandas, NLTK, Matplotlib. Дані бібліотеки є відкритими і комерційне використання не потребує додаткових витрат або дозволів.

Середовищем розробки було вибрано середовище VisualStudio Code за його простоту у використанні і наявність всього необхідного функціоналу для розробки даного програмного продукту. Оскільки програмний продукт реалізовано мовою Python, він є кросплатформеним і легко інтегруються як додаток в більшість сучасних сервісів.

Нижче наведено аналіз різних варіацій реалізації програмного продукту, для вибору найоптимальнішого, з використання функціонально-вартісного аналізу.

4.1 Обґрунтування функцій програмного продукту

Основними функціями програмного продукту є:

1. F_1 – метод розв’язку задачі ідентифікації.
2. F_2 – джерело бази даних повідомлень.
3. F_3 – використання алгоритмів.

Кожна з функцій може мати декілька варіантів реалізації.

Функція F_1:

а) нейронна мережа;

б) класифікатор.

Функція F_2:

а) взяття готових баз з інтернету;

б) формування власної бази даних.

Функція F_3:

а) власної розробки;

б) готові бібліотеки.

З наведених вище функцій складемо морфологічну карту (рисунок 4.1).

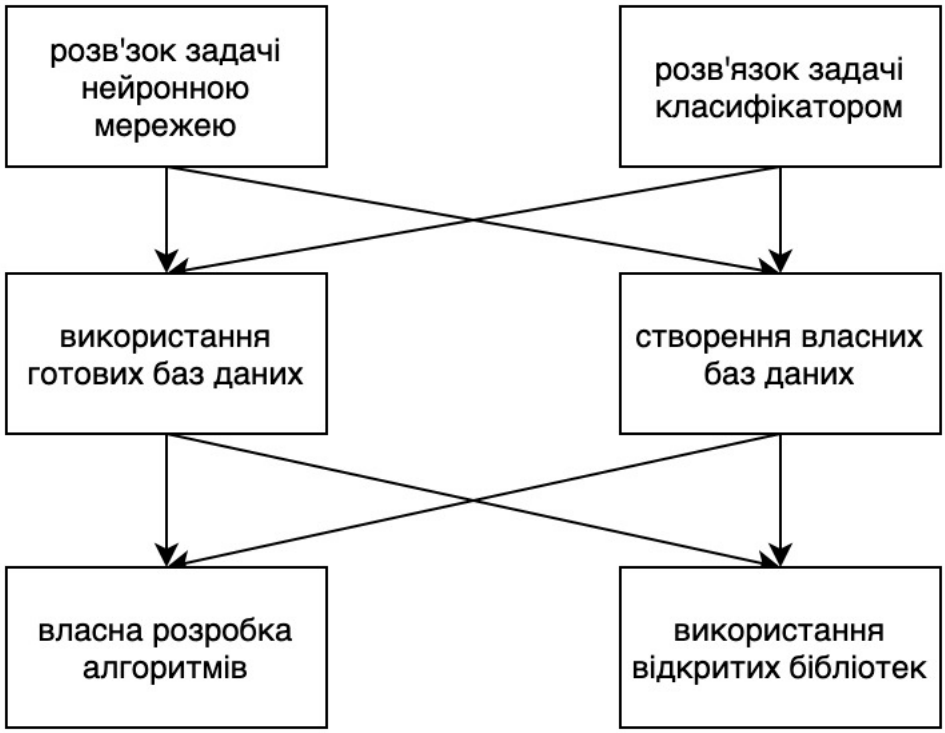

Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП. На основі морфологічної карти складемо позитивно-негативну матрицю (таблиця 4.1).

Таблиця 4.1 – Позитивно-негативна матриця

Основна функція	Варіант реалізації	Переваги	Недоліки
F1	А	Точність класифікації краща на приблизно 1%	Складність реалізації, час навчання моделі в декілька тисяч раз більший ніж у класифікатора
	Б	Швидкість навчання, відносна простота реалізації	Певні проблеми при появі нового слова при класифікації, якого не було при навчанні.
F2	А	Відносна простота в пошуку	Всі доступні бази даних англійською мовою
	Б	Можна реалізувати будь якою мовою	Займає значно більше часу ніж пошук готових, в Україні процес складання такої бази практично неможливий
F3	А	Трохи менше навантаження на систему за рахунок роботи тільки з потрібними функція	Значно більша складність реалізації, в рази більша кількість час на реалізацію
	Б	Економія людських ресурсів для реалізації за рахунок простоти, реалізований алгоритм найоптимальніший	Не така точність настройки всіх параметрів під конкретну поставлену задачу

За результатами аналізу таблиці залишимо наступні варіанти:

F_1 А – F_2А – F_3Б

F_1Б – F_2А – F_3Б

4.2 Обґрунтування системи параметрів ПП

4.2.1 Опис параметрів

Для того щоб охарактеризувати програмний продукт, використаємо наступні параметри:

- X1 – швидкість навчання та виконання ідентифікації програмою, тобто іншими словами швидкодія програми;
- X2 – час необхідний для реалізації моделі, тобто час затрачений людьми для написання програмного продукту і виконання необхідних розрахунків;
- X3 – час необхідний для ознайомлення з теорією, що в свою чергу необхідна при виконанні деяких розрахунків для реалізації програмного продукта;
- X4 – об’єм бази даних, тобто кількість повідомлення для навчання і тестування моделі.

Функція F_1 залежить від параметрів X1, X2, X3, тобто швидкість навчання, час необхідний для реалізації моделі та час необхідний для ознайомлення з теорією.

Функція F_2 залежить від параметрів X2, X4: час необхідний для реалізації моделі та кількість записів в базі даних.

Функція F_3 залежить від параметрів X2, X3: час необхідний для реалізації моделі і час необхідний для ознайомлення з теорією.

4.2.2 Кількісна оцінка параметрів

Будуємо таблицю 4.2 з параметрами.

Таблиця 4.2 – Система програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкість навчання та виконання ідентифікації програмою	X1	мс	10000	5000	50
Час, необхідний для реалізації моделі	X2	год	30	15	10
Час необхідний для ознайомлення з теорією	X3	год	100	70	40
Об’єм бази даних повідомлень	X4	шт	2000	4000	6000

Використовуючи дані таблиці побудуємо графіки (рисунок 4.2, рисунок 4.3, рисунок 4.4).

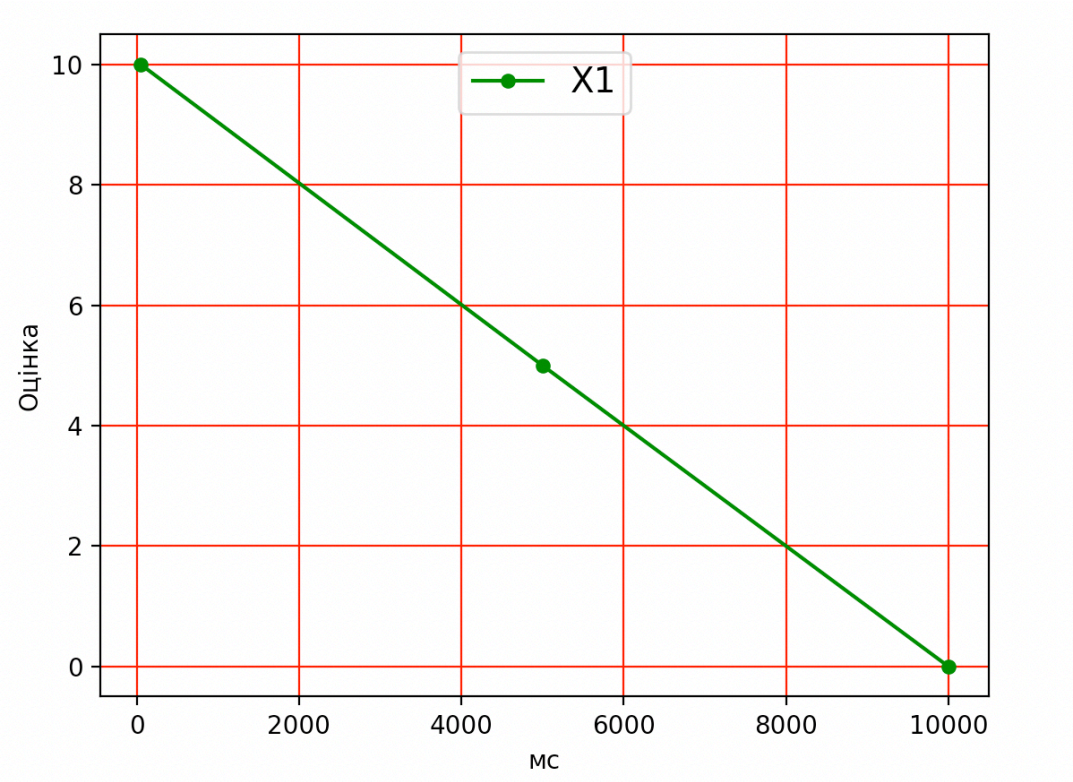

Рисунок 4.2 Значення параметру X1

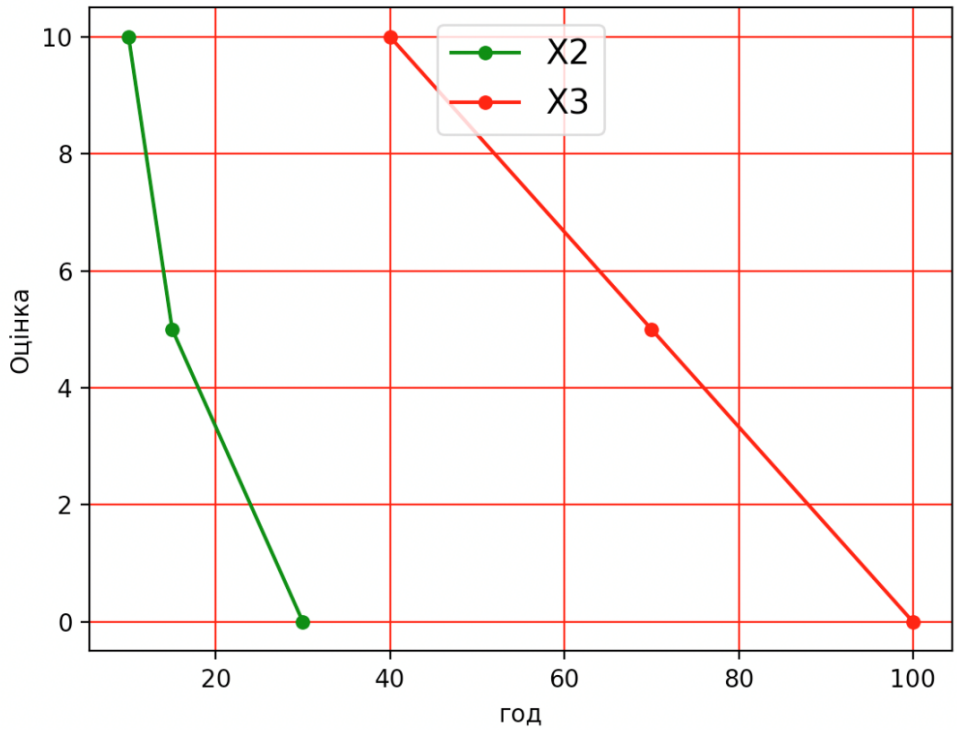

Рисунок 4.3 Значення параметрів X2 та X3

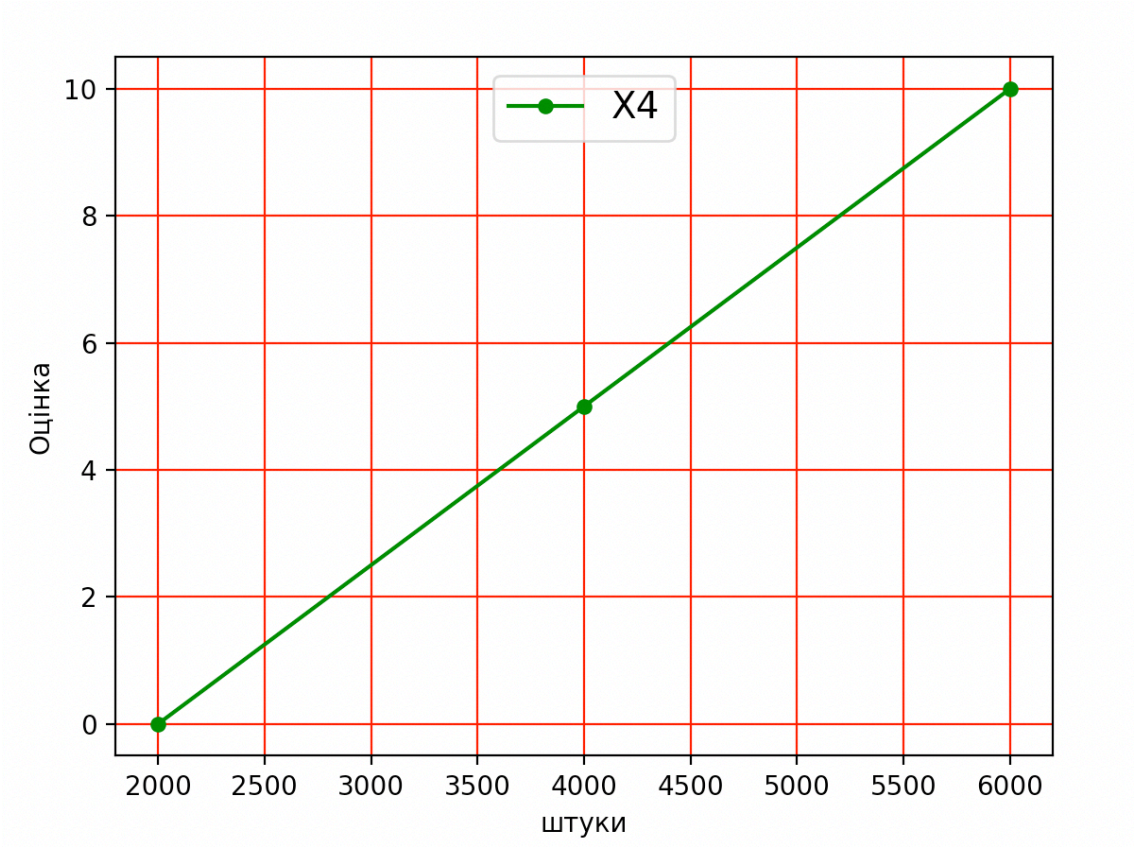

Рисунок 4.4 Значення параметру X4

Оцінимо параметри за допомогою метода попарного зрівняння (таблиця 4.3). В моєму випадку ранги варіюються від 1 до 4.

Таблиця 4.3 – Результат оцінки параметрів

Назва параметра	Умовні позначення	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
Швидкість навчання та виконання ідентифікації програмою	X1	мс	3	4	3	4	4	4	4	26	8.5	72.25
Час, необхідний для реалізації моделі	X2	год	4	3	4	2	3	3	3	22	4.5	20.25
Час необхідний для ознайомлення з теорією	X3	год	1	1	2	1	1	1	2	9	-8.5	72.25
Об’єм бази даних повідомлень	X4	шт	2	2	1	3	2	2	1	13	-4.5	20.25
	Разом		10	10	10	10	10	10	10	70	0	185

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \quad (4.1)$$

де N – число експертів;

n – кількість параметрів.

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17.5 \quad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \quad (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 185 \quad (4.4)$$

Коефіцієнт узгодженості дорівнює:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 * 185}{49(64 - 4)} = 0.755 > 0.67 \quad (4.5)$$

Експерти виставляли ранги від 1 до 4, тобто 1 це найменш важливий на думку експерта параметр, а 4 – найважливіший. Складемо таблицю попарного зрівняння параметрів (таблиця 4.4).

Таблиця 4.4 – Попарне зрівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	>	<	>	>	>	>	>	1.5
X1 і X3	>	>	>	>	>	>	>	>	1.5
X1 і X4	>	>	>	>	>	>	>	>	1.5
X2 і X3	>	>	>	>	>	>	>	>	1.5
X2 і X4	>	>	>	<	>	>	>	>	1.5
X3 і X4	<	<	>	<	<	<	>	<	0.5

Числове значення, що визначає ступінь переваги і–го параметра над j–тим, аіj визначається по формулі:

$$a_{ij} = \begin{cases} 1.5, \text{при Xi} > \text{Xj} \\ 1.0, \text{при Xi} = \text{Xj} \\ 0.5, \text{при Xi} < \text{Xj} \end{cases} \quad (4.6)$$

Використовуючи результати попарного порівняння обчислюється вагомість кожного з критеріїв (таблиця 4.5).

Таблиця 4.5 – Розрахунок вагомості параметрів

Xi	Параметри Xj,				Перший крок		Другий крок	
	X1	X2	X3	X4	$в_i$	$K_{вi}$	$в'_i$	$K'_{вi}$
X1	1.0	1.5	1.5	1.5	5.5	0.344	21.25	0.36
X2	0.5	1.0	1.5	1.5	4.5	0.281	16.25	0.275
X3	0.5	0.5	1.0	0.5	2.5	0.156	9.25	0.157
X4	0.5	0.5	1.5	1.0	3.5	0.218	12.25	0.208
Всього					16	1	59	1

Після другого кроку бачимо що зміна в відхиленні не перевищує 5%, тому можемо закінчити.

4.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо. Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}, \quad (4.7)$$

де n – кількість параметрів;

$K_{вi}$ – коефіцієнт вагомості і–го параметра;

B_i – оцінка і–го параметра в балах.

Таблиця 4.6 – Розрахунок показників якості варіантів реалізації функцій ПП

Основна функція	Варіант реалізації	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт якості
F1	А	X1	9000	1	0.36	0.36
		X2	20	3.33	0.275	0.916
		X3	50	8.33	0.157	1.31
F1	Б	X1	60	9.98	0.36	3.59
		X2	15	5	0. 275	1.375
		X3	40	10	0.157	1.57
F2	А	X2	12	8	0.275	2.2
		X4	5500	8.75	0.208	1.82
F3	Б	X2	12	8	0.275	2.2
		X3	40	10	0.157	1.57

$$K_1 = 0.36 + 0.916 + 1.31 + 2.2 + 1.82 + 2.2 + 1.57 = 10.376$$

$$K_2 = 3.59 + 1.375 + 1.57 + 2.2 + 1.82 + 2.2 + 1.57 = 14.325$$

Як бачимо, варіант 2 має більший коефіцієнт якості, отже він є кращим.

4.4 Економічний аналіз варіантів розробки ПП

Варіанти реалізації продукту можна описати наступним чином:

1. Пошук бази даних;

2. Створення моделі:

А) Створення та навчання нейронної мережі;

Б) Створення та навчання класифікатора;

3. Демонстрація результатів.

Для завдання 1 маємо: алгоритм групи складності 3, ступінь новизни Г, вид використаної інформації БД, $Т_р = 8, К_п = 0.3, К_{ск} = 1.16, К_{ст.м} = 1.2$

$$Т_1 = 8 \cdot 0.3 \cdot 1.16 \cdot 1.2 = 3{,}34 \text{ людино-днів.}$$

Для завдання 2 (варіант А) маємо: алгоритм групи складності 1, ступінь новизни Б, вид використаної інформації ПІ, $Т_р = 64, К_п = 2.02, К_{ск} = 1.08, К_{ст.м} = 1.5$

$$Т_{2а} = 64 \cdot 2.02 \cdot 1.08 \cdot 1.5 = 209{,}434 \text{ людино-днів.}$$

Для завдання 2 (варіант Б) маємо: алгоритм групи складності 1, ступінь новизни Б, вид використаної інформації ПІ, $Т_р = 64, К_п = 2.02, К_{ск} = 1.08, К_{ст.м} = 1.35$

$$Т_{2б} = 64 \cdot 2.02 \cdot 1.08 \cdot 1.35 = 188{,}49 \text{ людино-днів.}$$

Для завдання 3маємо: алгоритм групи складності 2, ступінь новизни Г, вид використаної інформації БД, $Т_р = 12, К_п = 0.36, К_{ск} = 1.07, К_{ст.м} = 1.2$

$$Т_3 = 12 \cdot 0.36 \cdot 1.07 \cdot 1.2 = 5{,}547 \text{ людино-днів.}$$

Тоді для варіанту А:

$$\text{Т} = (3{,}34 + 209.434 + 5.547) \cdot 8 = 1746.568 \text{ людино-годин}$$

а для варіанту Б:

$$\text{Т} = (3{,}34 + 188.49 + 5.547) \cdot 8 = 1579.016 \text{ людино-годин}$$

В розробці бере участь один програміст з окладом 14000 грн та один математик з окладом 12000 грн.

Розрахунок середньої заробітної плати за годину:

$$\text{С} = \frac{26000}{2*22*8} = 73.864 \text{ грн} \quad (4.8)$$

Заробітна плата для кожного з варіантів реалізації

$$\text{С}_1 = \ 73.864 * 1746.568 = 129008{,}5 \quad (4.9)$$

$$\text{С}_2 = \ 73.864 * 1579.016 = 116632{,}438 \quad (4.10)$$

Відрахування ЄСВ складають 22%, або для кожного варіанту:

$$\text{С}_{\text{від1}} = \ 0.22 * 129008{,}5 = 28381{,}87$$

$$\text{С}_{\text{від2}} = 0.22 * 116632{,}438 = 25659{,}136$$

Визначимо виплату необхідну на оплату однієї машино-години.

Враховуючи, що заробітна плата одного програміста складає 14000 грн з коефіцієнтом зайнятості 0.4

$$С_г = 12 * М * К_з = 12 * 14000 * 0.4 = 67200 \text{ грн}$$

Враховуючи додаткову заробітну плату:

$$С_{зп} = С_г \cdot (1 + К_з) = 67200 \cdot 1.4 = 94080 \text{ грн} \quad (4.11)$$

Відрахування на соціальний внесок складатимуть:

$$С_{від} = 0,22 \cdot 94080 = 20697.6 \text{ грн} \quad (4.12)$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 18000 грн.

$$С_А = К_{тм} \cdot К_а \cdot Ц_{пр} = 1.15 \cdot 0.25 \cdot 18000 = 5175 \text{ грн}$$

Розрахуємо витрати на профілактику та ремонт:

$$С_р = К_{тм} \cdot К_р \cdot Ц_{пр} = 1.15 \cdot 0.05 \cdot 18000 = 1035 \text{ грн}$$

Розраховуємо ефективний годинний фонд часу ПК за рік:

$$Т_{ЕФ} = (365 - 104 - 11 - 4) \cdot 8 \cdot 0.8 = 1574,4 \text{год}$$

Розраховуємо витрати на електроенергію:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1574,4 \cdot 0,28 \cdot 0,9 \cdot 1,75 = 694,31 \text{ грн}$$

Розрахуємо накладні витрати:

$$С_Н = 18000 \cdot 0.67 = 12060 грн$$

Розраховуємо річні експлуатаційні витрати:

$$С_{ЕКС} = 94080 + 20697.6 + 5175 + 1035 + 694,31 = 121681,91 \text{ грн} \quad (4.13)$$

Розраховуємо собівартість однієї машино-години

$$С_{М-Г} = \frac{121681,91}{1574,4} = 77.29 \text{ грн/год} \quad (4.14)$$

Розраховуємо оплату машинного часу в залежності від варіантів:

$$С_{М1} = 77.29 \cdot 1746.568 = 134988.396 \text{ грн} \quad (4.15)$$

$$С_{М2} = 77.29 \cdot 1579.016 = 122038.67 \text{ грн} \quad (4.16)$$

Розраховуємо накладні витрати в залежності від варіантів:

$$С_{Н1} = 129008.5 \cdot 0.67 = 86435.695 \text{ грн} \quad (4.17)$$

$$С_{Н2} = 116632.438 \cdot 0.67 = 78143.733 \text{ грн} \quad (4.18)$$

Розраховуємо вартість розробки в залежності від варіантів:

$$С_{ПП1} = 129008.5 + 28381,87 + 134988.396 + 86435.695 = 378814.46 \text{ грн}$$

$$С_{ПП2} = 116632,438 + 25659,136 + 122038.67 + 78143.733 = 342473.978 \text{ грн}$$

4.5 Вибір кращого варіанта ПП за техніко-економічного рівня

Розраховуємо коефіцієнти техніко-економічного рівня для кожного варіанта за формулою:

$$К_{ТЕРj} = К_{Кj}/С_{Фj} \quad (4.19)$$

Для першого варіанта:

$$К_{ТЕР1} = 10.376/378814.46 = 2.739 \cdot 10^{-5} \quad (4.20)$$

Для другого варіанта:

$$К_{ТЕР2} = 14.325/342473.978 = 4.182 \cdot 10^{-5} \quad (4.21)$$

Отже, проаналізувавши результати робимо висновок, що найефективнішим буде варіант з коефіцієнтом техніко-економічного рівня $4.182 \cdot 10^{-5}$.

4.6 Висновки до розділу 4

Після проведеного функціонально-вартісного аналізу було прийняте рішення про розробку програмного продукту з використанням класифікатора, існуючої бази даних спам повідомлень та готових відкритих бібліотек для роботи з нею.

ВИСНОВКИ ПО РОБОТІ

Дана робота була присвячена задачі ідентифікації спаму та її вирішенню методом наївного баєсового класифікатора.

В роботі також було проведено порівняльний аналіз всіх актуальних методів для ідентифікації спаму, в результаті якого було виявлено найефективніший метод для вирішення задачі класифікації спаму.

В рамках даної бакалаврської роботи були отримані результати, які показують що наївний баєсовий класифікатор Бернуллі найкращий для ідентифікації спаму. Було розроблено програмний продукт, що робить аналіз дата-сету та розпізнає спам повідомлення, з використанням відкритих бібліотек на мові програмування Python.

Покращити дану роботу можна помінявши трохи код програмного продукту для динамічного навчання, тобто в реальному часі коли йде потік даних і програма динамічно підлаштовується під нові дані і таким чином видає більшу точність. Також можна спробувати переписати даний програмний продукт іншою мовою програмування, наприклад C++, і можливо таким чином зменшиться час на навчання та ідентифікацію.

ПЕРЕЛІК ПОСИЛАНЬ

1. More Than 90% of E-Mails Were Spam. URL: http://certmag.com/more-than-90-percent-of-e-mails-in-third-quarter-were-spam/ (Last accessed: 10.05.2020)
2. Zhang H. The Optimality of Naive Bayes. URL: https://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf (Last accessed: 11.05.2020)
3. Naive Bayes. URL: https://scikit-learn.org/stable/modules/naive_bayes.html (Last accessed: 14.05.2020)
4. N. Friedman, D. Geiger, and M. Goldszmidt Bayesian network classifiers. Machine Learning. 1997. vol. 29, P. 131–163.
5. Eric B. Baum On the Capabilities of Multilayer Perceptrons. *Journal of Complexity*. 1988. No 4. P. 193-215.
6. SMS Spam Collection URL: https://www.researchgate.net/publication/258050002_SMS_Spam_Collection_v1 (Last accessed: 15.05.2020)
7. Теорія імовірностей та математична статистика. Курс лекцій. / Уклад.: Т.А.Ліхоузова – К.: КПІ ім. Ігоря Сікорського, 2018. – 300 с.
8. ДСТУ 3008:2015. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. [На заміну ДСТУ 3008-95; чинний від 2015-06-22]. Вид. офіц. Київ: ДП «УкрНДНЦ», 2016. 31 с. (Інформація та документація).

ДОДАТОК А

Лістинг програми для поліноміального наївного баєсового класифікатора

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import time
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from nltk.stem import SnowballStemmer
from wordcloud import WordCloud
import warnings
warnings.filterwarnings('ignore')

import nltk
from nltk.corpus import stopwords
import string
from nltk.tokenize import word_tokenize

import spacy

nlp = spacy.load("en")
#зчитування дата-сету
messages = pd.read_csv("spam.csv", encoding = 'utf-8')

#побудова графіка співвідношеня повідомлень
```

```
messages["category"].value_counts().plot(kind = 'pie', explode = [0, 0.01], autopct
= '%1.2f%%', shadow = False, colors=["green", "red"], fontsize=23)
plt.ylabel("Статистика")
plt.legend(["Звичайне", "Спам"])
plt.show(block = False)

spam_messages = messages[messages["category"] == "spam"]["text"]
ham_messages = messages[messages["category"] == "ham"]["text"]

#виділення з повідомлень слів для подальшої обробки
spam_words = []
ham_words = []
def extractSpamWords(spamMessages):
    global spam_words
    words = [word.lower() for word in word_tokenize(spamMessages) if
word.lower() not in stopwords.words("english") and word.lower().isalpha()]
    spam_words = spam_words + words

def extractHamWords(hamMessages):
    global ham_words
    words = [word.lower() for word in word_tokenize(hamMessages) if
word.lower() not in stopwords.words("english") and word.lower().isalpha()]
    ham_words = ham_words + words
spam_messages.apply(extractSpamWords)
ham_messages.apply(extractHamWords)

#будуються картинки з зображення найбільш часто вживаних слів
spam_wordcloud = WordCloud(width=600, height=400,
background_color="red").generate(" ".join(spam_words))
plt.figure( figsize=(10,8), facecolor='w')
```

```
plt.imshow(spam_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
ham_wordcloud = WordCloud(width=600, height=400,
background_color="green").generate(" ".join(ham_words))
plt.figure( figsize=(10,8), facecolor="green")
plt.imshow(ham_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()

# виводяться графіки з довжинами повідомлень
messages["messageLength"] = messages["text"].apply(len)
sns.distplot(messages[messages["category"] == "spam"]["messageLength"], bins =
20, color="red")
plt.xlabel("Довжина спам повідомлень", fontsize=14)
plt.show()
sns.distplot(messages[messages["category"] == "ham"]["messageLength"], bins =
20, color="green")
plt.xlabel("Довжина звичайних повідомлень", fontsize=14)
plt.show()

#відтинаються суфікси і префікси і слова, не несущі смислової нагрузки
stemmer = SnowballStemmer("english")

def cleanText(message):
    message = message.translate(str.maketrans(", ", string.punctuation))
    words = [stemmer.stem(word) for word in message.split() if word.lower() not in
stopwords.words("english")]
```

```
    return " ".join(words)
messages["text"] = messages["text"].apply(cleanText)
messages.head(n = 10)

#перетворюємо повідомлення в tf-idf вектор
vec = TfidfVectorizer(encoding = "utf-8", strip_accents = "unicode", stop_words =
"english")
features = vec.fit_transform(messages["text"])

def encodeCategory(sort):
    if sort == "spam":
        return 1
    else:
        return 0
messages["category"] = messages["category"].apply(encodeCategory)

#Розбиваємо вибірку на навчальну і тестову
X_train, X_test, y_train, y_test = train_test_split(features, messages["category"],
stratify = messages["category"], test_size = 0.25)
time.sleep(10)

#тест поліноміального наївного баєсового класифікатора
start = time.time()
naive_bayes_model = MultinomialNB(alpha = 0)
naive_bayes_model.fit(X_train, y_train)
y_pred =  naive_bayes_model.predict(X_test)
end = time.time()

fin=y_test.to_frame()
fin = fin["category"].tolist()
```

```
p = 0
for i in range(len(y_pred)):
        if(y_pred[i]==fin[i]):
            p +=1
        #print(y_pred[i], fin[i])
print("Поліноміальний наївний баєсовий класифікатор\nТочність моделі: ",
p/len(y_pred))
print("Час затрачений на навчання та прогнозування: ",end-start)
k = naive_bayes_model.predict_proba(X_test)
k = k.T
plt.plot(k[0][:70],marker='o', markersize=12, color = "green")
plt.plot(k[1][:70], color = "orange",marker='o', markersize=12)
```

ДОДАТОК Б

Лістинг програми для наївного баєсового класифікатора Бернулі

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import time
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import BernoulliNB
from nltk.stem import SnowballStemmer
from wordcloud import WordCloud
import warnings
warnings.filterwarnings('ignore')
import nltk
from nltk.corpus import stopwords
import string
from nltk.tokenize import word_tokenize
import spacy

nlp = spacy.load("en")
#зчитування дата-сету
messages = pd.read_csv("spam.csv", encoding = 'utf-8')

#побудова графіка співвідношеня повідомлень
messages["category"].value_counts().plot(kind = 'pie', explode = [0, 0.01], autopct
= '%1.2f%%', shadow = False, colors=["green", "red"], fontsize=23)
plt.ylabel("Статистика")
```

```
plt.legend(["Звичайне", "Спам"])
plt.show(block = False)

#виділення з повідомлень слів для подальшої обробки

spam_messages = messages[messages["category"] == "spam"]["text"]
ham_messages = messages[messages["category"] == "ham"]["text"]
spam_words = []
ham_words = []
def extractSpamWords(spamMessages):
    global spam_words
    words = [word.lower() for word in word_tokenize(spamMessages) if
word.lower() not in stopwords.words("english") and word.lower().isalpha()]
    spam_words = spam_words + words

def extractHamWords(hamMessages):
    global ham_words
    words = [word.lower() for word in word_tokenize(hamMessages) if
word.lower() not in stopwords.words("english") and word.lower().isalpha()]
    ham_words = ham_words + words
spam_messages.apply(extractSpamWords)
ham_messages.apply(extractHamWords)

#будуються картинки з зображення найбільш часто вживаних слів
spam_wordcloud = WordCloud(width=600, height=400,
background_color="red").generate(" ".join(spam_words))
plt.figure( figsize=(10,8), facecolor='w')
plt.imshow(spam_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
```

```
plt.show()
ham_wordcloud = WordCloud(width=600, height=400,
background_color="green").generate(" ".join(ham_words))
plt.figure( figsize=(10,8), facecolor="green")
plt.imshow(ham_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()

# виводяться графіки з довжинами повідомлень
messages["messageLength"] = messages["text"].apply(len)
sns.distplot(messages[messages["category"] == "spam"]["messageLength"], bins =
20, color="red")
plt.xlabel("Довжина спам повідомлень", fontsize=14)
plt.show()
sns.distplot(messages[messages["category"] == "ham"]["messageLength"], bins =
20, color="green")
plt.xlabel("Довжина звичайних повідомлень", fontsize=14)
plt.show()

#відтинаються суфікси і префікси і слова, не несущі смислової нагрузки
stemmer = SnowballStemmer("english")

def cleanText(message):
    message = message.translate(str.maketrans('', '', string.punctuation))
    words = [stemmer.stem(word) for word in message.split() if word.lower() not in
stopwords.words("english")]
    return " ".join(words)

messages["text"] = messages["text"].apply(cleanText)
```

```
messages.head(n = 10)

#перетворюємо повідомлення в tf-idf вектор
vec = TfidfVectorizer(encoding = "utf-8", strip_accents = "unicode", stop_words =
"english")
features = vec.fit_transform(messages["text"])

def encodeCategory(sort):
    if sort == "spam":
        return 1
    else:
        return 0
messages["category"] = messages["category"].apply(encodeCategory)

#Розбиваємо вибірку на навчальну і тестову
X_train, X_test, y_train, y_test = train_test_split(features, messages["category"],
stratify = messages["category"], test_size = 0.25)
time.sleep(10)

#тест наївного баєсового класифікатора Бернуллі
start = time.time()
naive_bayes_model = BernoulliNB()
naive_bayes_model.fit(X_train, y_train)
y_pred = naive_bayes_model.predict(X_test)
end = time.time()

fin=y_test.to_frame()
fin = fin["category"].tolist()
p = 0
```

```
for i in range(len(y_pred)):
        if(y_pred[i]==fin[i]):
            p +=1


print("Наївний баєсовий класифікатор Бернуллі\nТочність моделі: ",
p/len(y_pred))
print("Час затрачений на навчання та прогнозування: ",end-start)
k = naive_bayes_model.predict_proba(X_test)
k = k.T
plt.figure(figsize = (15, 10))
plt.plot(k[0][:70],marker='o', markersize=12,color = "green")
plt.plot(k[1][:70], color = "orange",marker='o', markersize=12)
plt.show()
```

ДОДАТОК В

Лістинг програми для багатошарового перцептрона

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import time
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import fbeta_score
from sklearn.neural_network import MLPClassifier
from nltk.stem import SnowballStemmer
from wordcloud import WordCloud
import warnings
warnings.filterwarnings('ignore')

import nltk
from nltk.corpus import stopwords
import string
from nltk.tokenize import word_tokenize

import spacy

nlp = spacy.load("en")
#зчитування дата-сету
messages = pd.read_csv("spam.csv", encoding = 'utf-8')

#побудова графіка співвідношеня повідомлень
```

```
messages["category"].value_counts().plot(kind = 'pie', explode = [0, 0.01], autopct
= '%1.2f%%', shadow = False, colors=["green", "red"], fontsize=23)
plt.ylabel("Статистика")
plt.legend(["Звичайне", "Спам"])
plt.show(block = False)

spam_messages = messages[messages["category"] == "spam"]["text"]
ham_messages = messages[messages["category"] == "ham"]["text"]

#виділення з повідомлень слів для подальшої обробки
spam_words = []
ham_words = []
def extractSpamWords(spamMessages):
    global spam_words
    words = [word.lower() for word in word_tokenize(spamMessages) if
word.lower() not in stopwords.words("english") and word.lower().isalpha()]
    spam_words = spam_words + words

def extractHamWords(hamMessages):
    global ham_words
    words = [word.lower() for word in word_tokenize(hamMessages) if
word.lower() not in stopwords.words("english") and word.lower().isalpha()]
    ham_words = ham_words + words
spam_messages.apply(extractSpamWords)
ham_messages.apply(extractHamWords)

#будуються картинки з зображення найбільш часто вживаних слів
spam_wordcloud = WordCloud(width=600, height=400,
background_color="red").generate(" ".join(spam_words))
plt.figure( figsize=(10,8), facecolor='w')
```

```
plt.imshow(spam_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
ham_wordcloud = WordCloud(width=600, height=400,
background_color="green").generate(" ".join(ham_words))
plt.figure( figsize=(10,8), facecolor="green")
plt.imshow(ham_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()

# виводяться графіки з довжинами повідомлень
messages["messageLength"] = messages["text"].apply(len)
sns.distplot(messages[messages["category"] == "spam"]["messageLength"], bins =
20, color="red")
plt.xlabel("Довжина спам повідомлень", fontsize=14)
plt.show()
sns.distplot(messages[messages["category"] == "ham"]["messageLength"], bins =
20, color="green")
plt.xlabel("Довжина звичайних повідомлень", fontsize=14)
plt.show()

#відтинаються суфікси і префікси і слова, не несущі смислової нагрузки
stemmer = SnowballStemmer("english")

def cleanText(message):
    message = message.translate(str.maketrans('', '', string.punctuation))
    words = [stemmer.stem(word) for word in message.split() if word.lower() not in
stopwords.words("english")]
```

```
    return " ".join(words)

messages["text"] = messages["text"].apply(cleanText)
messages.head(n = 10)

#перетворюємо повідомлення в tf-idf вектор
vec = TfidfVectorizer(encoding = "utf-8", strip_accents = "unicode", stop_words =
"english")
features = vec.fit_transform(messages["text"])

def encodeCategory(sort):
    if sort == "spam":
        return 1
    else:
        return 0

messages["category"] = messages["category"].apply(encodeCategory)

#Розбиваємо вибірку на навчальну і тестову
X_train, X_test, y_train, y_test = train_test_split(features, messages["category"],
stratify = messages["category"], test_size = 0.25)
time.sleep(10)

#тест багатошарового перцептрона
start = time.time()
clf = MLPClassifier(activation='relu',solver='adam', batch_size=50,alpha=1e-
5,hidden_layer_sizes=(150,140,130),
random_state=42,learning_rate='adaptive')
clf.fit(X_train, y_train)
y_predpa = clf.predict(X_test)
```

```
end = time.time()

p = 0
for i in range(len(y_pred)):
        if(y_predpa[i]==fin[i]):
            p +=1
print("Багатошаровий перцептрон\nТочність моделі: ", p/len(y_pred))
print("Час затрачений на навчання та прогнозування: ",end-start)

plt.plot(clf.loss_curve_)
plt.show()
```

ДОДАТОК Г

Ілюстративний матеріал

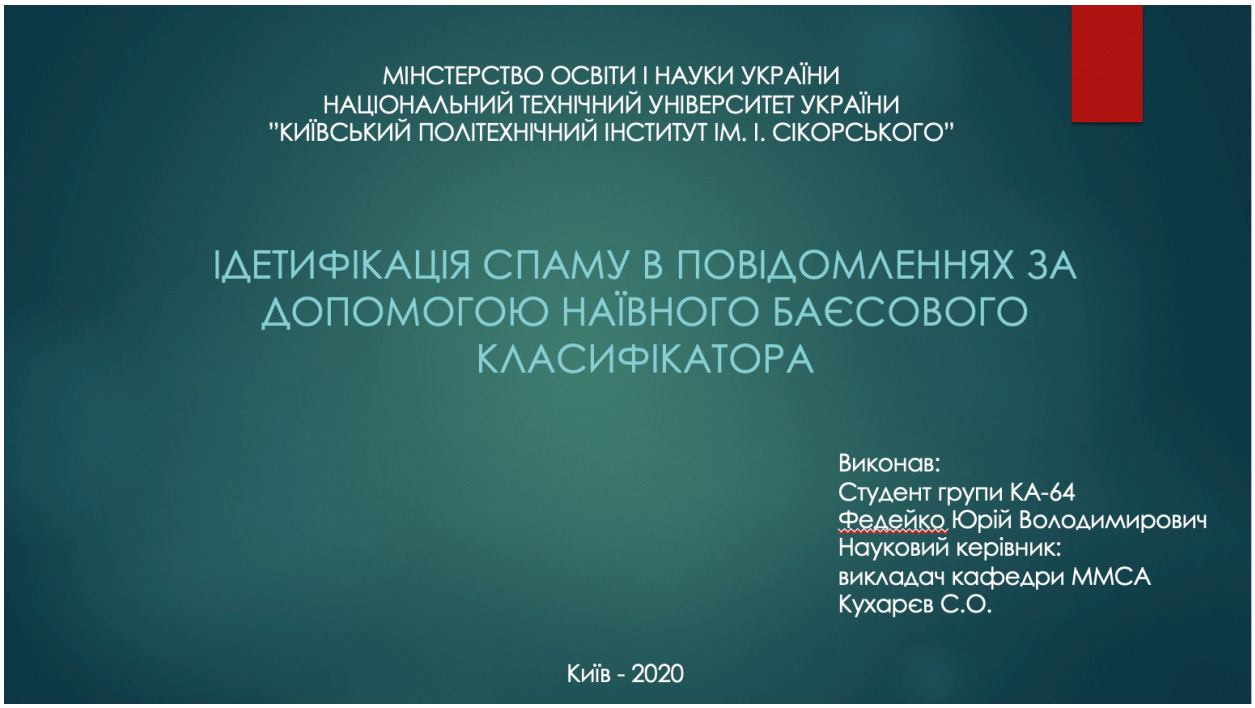

Рисунок 1 – Вступний слайд

Рисунок 2 – Актуальність теми

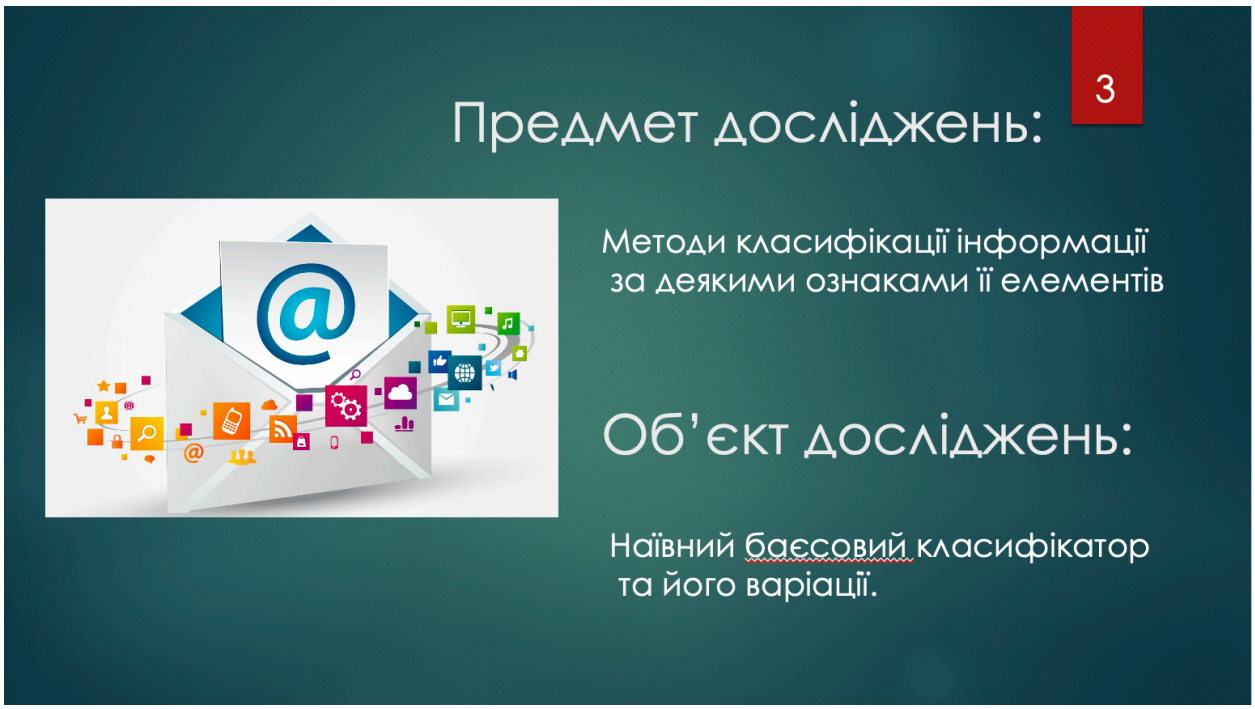

Рисунок 3 – Предмет та об'єкт досліджень

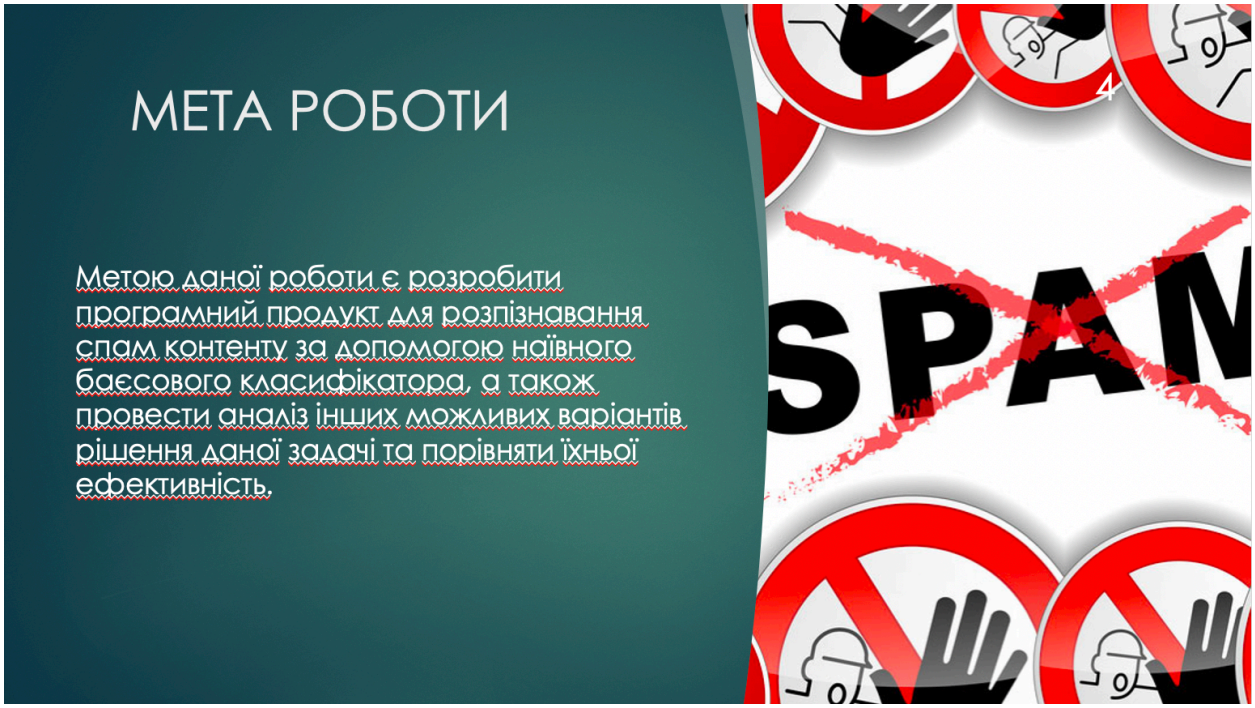

Рисунок 4 – Мета роботи

ПОСТАНОВКА ЗАДАЧІ

5

Маємо тестову вибірку з N елементів, в нашому випадку це база даних.

Кожне повідомлення характеризується m числових характеристик $a_1, \dots, a_m$.

Значення i-го признака j-го елемента, що належить k-му класу позначимо як x_{kji}.

$\mathrm{x}_{\mathrm{kj}} = (\mathrm{x}_{\mathrm{kj1}}, \dots, \mathrm{x}_{\mathrm{kji}}, \dots, \mathrm{x}_{\mathrm{kjm}})$

Тепер спостерігаємо об'єкт для якого необхідно визначити клас, тобто спам чи звичайне повідомлення.

Об'єкт характеризується тільки набором m числових ознак $\mathrm{x}_1, \dots, \mathrm{x}_{\mathrm{m}}$.

Далі ми передаєм ці дані в класифікатор, і знаходимо певну "прив'язаність" слова до деякого класу.

Рисунок 5 – постановка задачі

ОПИС АКТУАЛЬНИХ МЕТОДІВ

6

Першим є поліноміальний наївний баєсів класифікатор. Він зазвичай використовується для класифікації документів з подіями, що означають частоту появи слова в одному документі. Апостеріорна ймовірність тоді розраховується за формулою:

$$p(\mathbf{x} \mid C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}^{x_i} \qquad (1)$$

Рисунок 6 – Опис поліноміального наївного баєсового класифікатора

ОПИС АКТУАЛЬНИХ МЕТОДІВ

7

Другим хорошим методом для даної задачі є наївний баєсів класифікатор Бернуллі. Він використовується також для класифікації документів, але тільки з двома класами, що в моєму випадку підходить для задачі.

$$p(\mathbf{x} \mid C_k) = \prod_{i=1}^{n} p_{ki}^{x_i} (1 - p_{ki})^{(1-x_i)} \qquad (2)$$

Рисунок 7 – Опис наївного баєсового класифікатора Бернулі

ОПИС АКТУАЛЬНИХ МЕТОДІВ

8

Нейронні межі зараз дуже популярні і їх використовують для вирішення дуже великого роду задач.

Для прикладу була обрана одна з найпростіших нейронних мереж – багатошаровий перцептрон з трьома шарами прихованих нейронів.

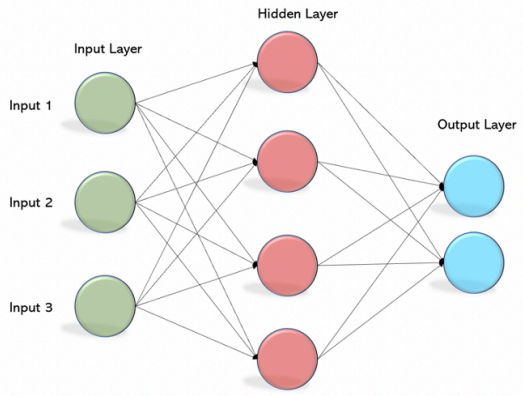

Рисунок 8 – Опис багатошарового перцептрона

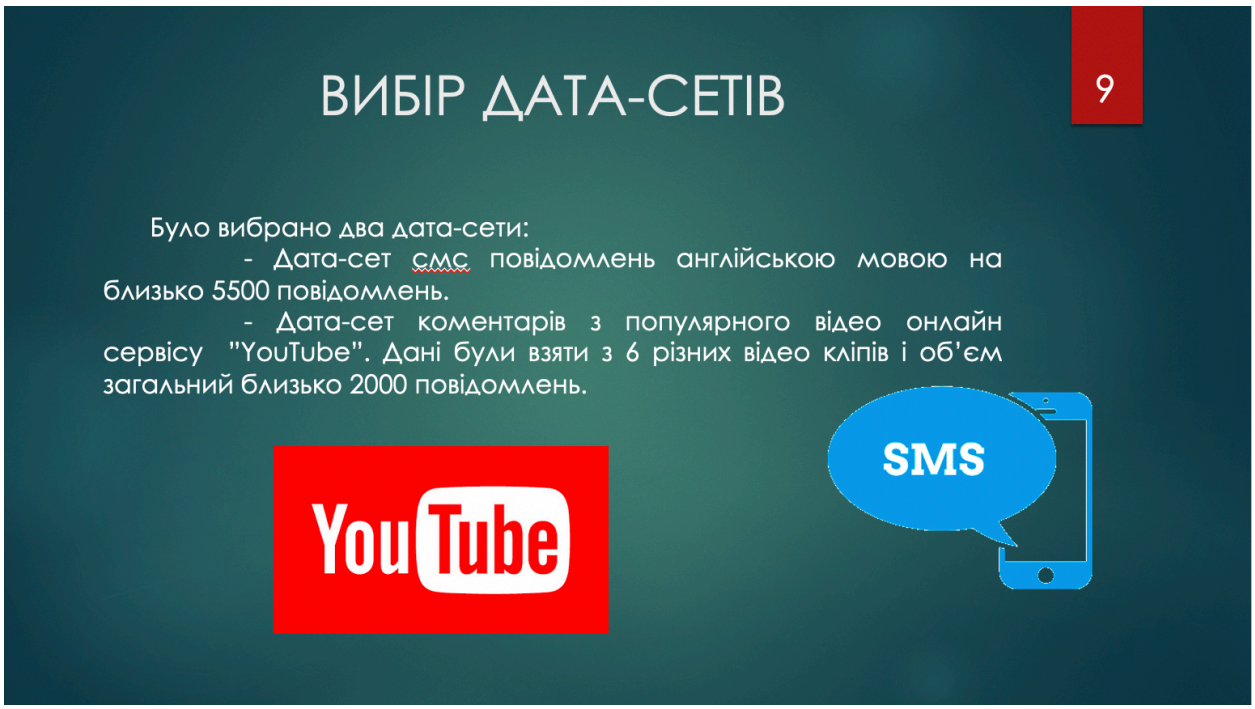

Рисунок 9 – Вибір дата-сетів

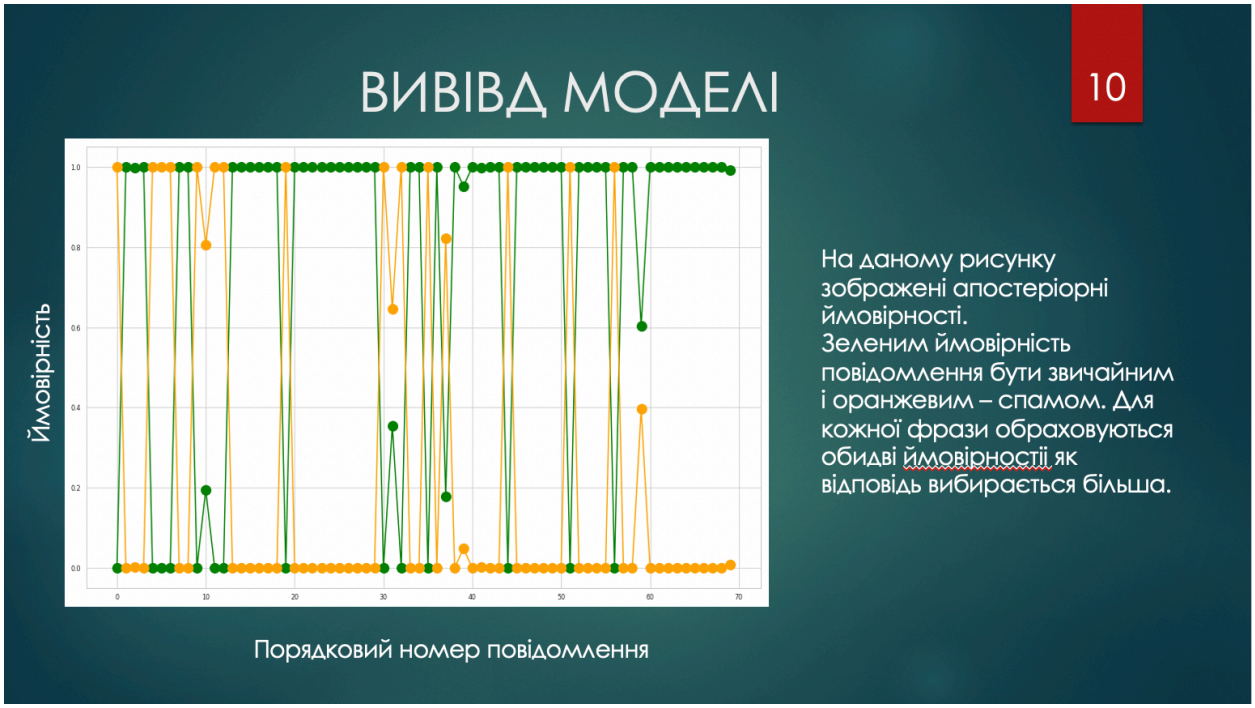

Рисунок 10 – Вивід моделі

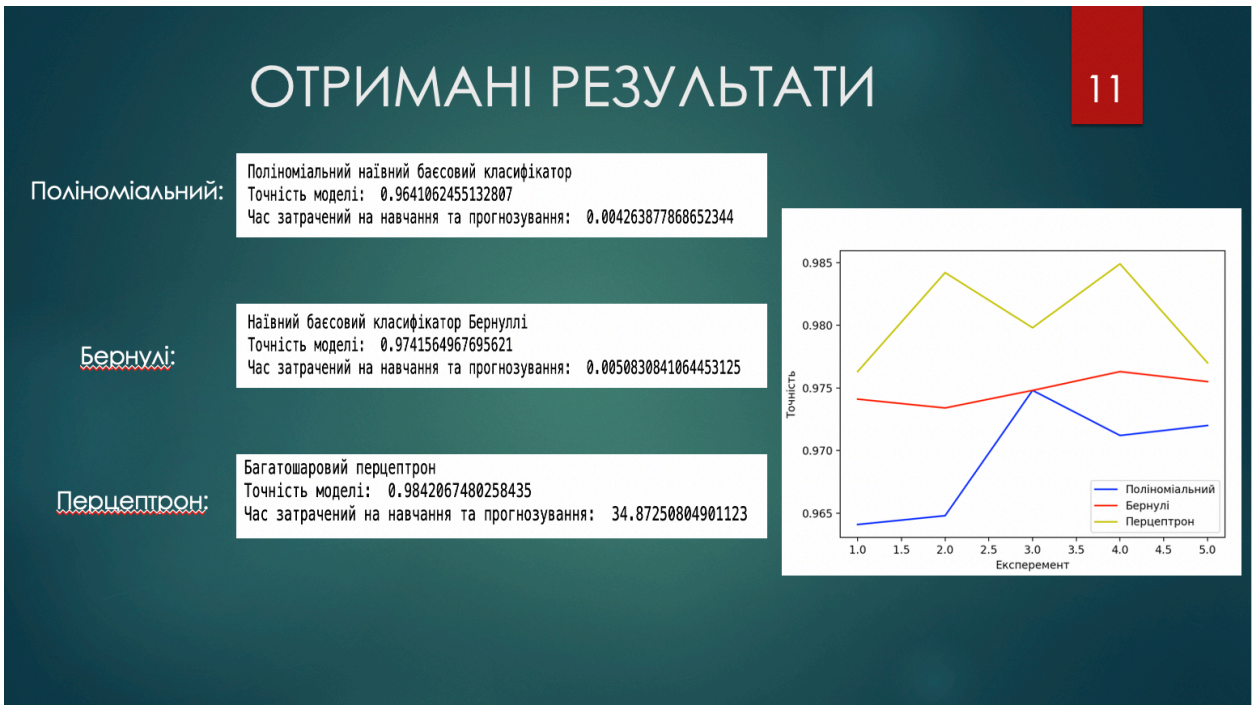

Рисунок 11 – Отримані результати

Рисунок 12 – Способи вдосконалення роботи

Рисунок 13 – Висновки

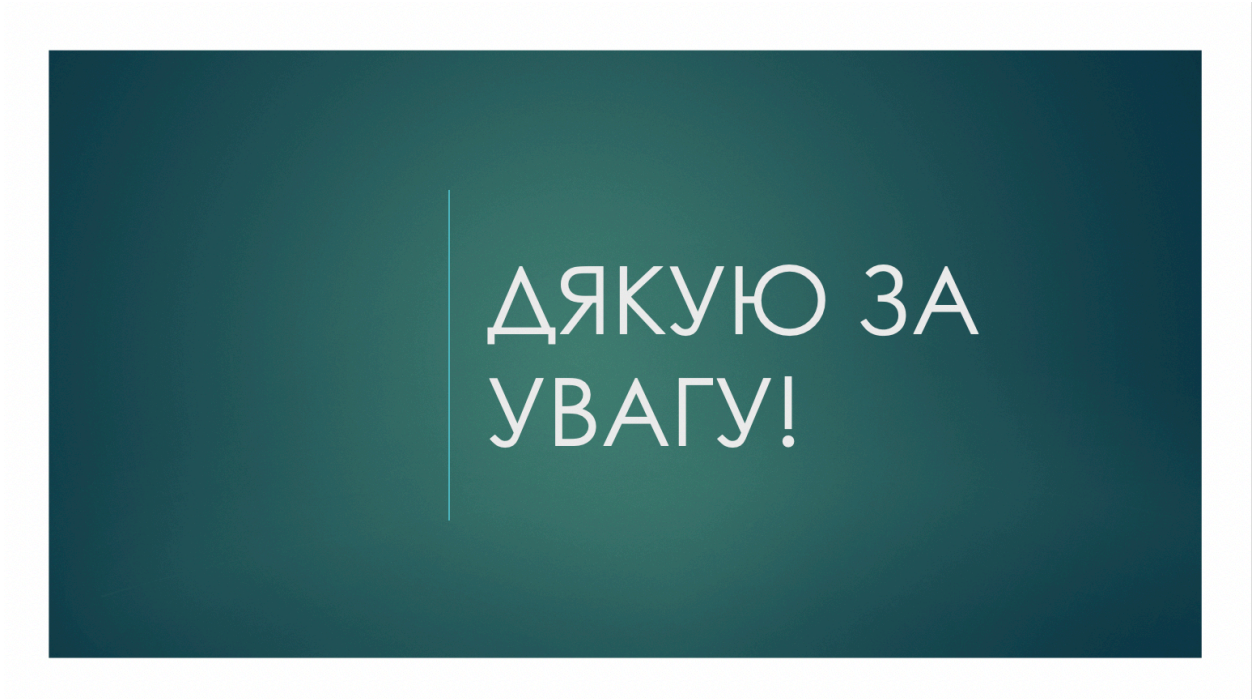

Рисунок 14 – Завершальний слайд